INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia

de Segurança Informática

Preventive Vulnerability Scanner

Sistema de detecção de vulnerabilidade em aplicações instaladas em sistemas *Windows*

Pedro Xavier Monteiro Godinho

2016

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia

de Segurança Informática

Preventive Vulnerability Scanner

Sistema de detecção de vulnerabilidade em aplicações instaladas em sistemas *Windows*

Relatório de Dissertação de Mestrado apresentado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

Elaborado por:

Pedro Xavier Monteiro Godinho

Orientado por:

Professor Dr. Rui Miguel Soares Silva

2016

Resumo

Preventive Vulnerability Scanner

Nos dias de hoje, a segurança informática representa uma preocupação prioritária para as organizações. Esta crescente importância da segurança no seio das organizações deve-se, em grande parte, à divulgação de um número crescente de ataques informáticos de enorme relevância, recorrendo cada vez mais a tecnologias inovadoras para a realização dos mesmos.

Os ataques a sistemas informáticos, envolvem normalmente, uma pesquisa sobre vulnerabilidades conhecidas e sobre o sistema utilizado pela vítima, utilizando uma ligação à Internet ou através de uma rede informática. Este conjunto de dados fornecem, aos atacantes, informações que possibilitam a exploração de uma falha associada às características de um sistema.

Para combater este tipo de ameaças informáticas, existem aplicações cujo principal objetivo é a deteção de falhas em sistemas informáticos e nas aplicações presentes nos mesmos. Estas aplicações, verificam o nível de conformidade de um sistema através da comparação do estado atual do sistema com uma lista de vulnerabilidades, configurações e correções padronizadas, permitindo assim a deteção de falhas.

Esta Dissertação de Mestrado, realizada em contexto de estágio, tratará do desenvolvimento de uma aplicação para a deteção de vulnerabilidades e a sua integração numa plataforma de *Ciberintelligence*, recorrendo a ferramentas e tecnologias *open source*.

Palavras-chave: *Segurança Informática, Ataque Informático, Deteção de Vulnerabilidades, Desenvolvimento de Software, Sistemas Operativos Windows.*

Abstract

Preventive Vulnerability Scanner

Now a day, computer security is a priority concern. This growing importance of security within organizations, is due to the dissemination of a growing number of computer attacks of enormous relevance, that use a lot of innovative technologies for their realization.

Attacks on computer systems usually involve a lot of research into known vulnerabilities and the system used by the victim, using an internet connection or through a computer network. This set of data provides attackers with information that enables them to exploit a fault associated with the characteristics of a system.

To combat this type of computer threats, there are applications whose main objective is the detection of failures in a computer system as also the software present in them. These applications check the conformity level of a system by comparing the current state of the system with a list of vulnerabilities, configurations and standardized patches, thus allowing for the detection of failures.

This Masters Dissertation, carried out in an internship context, will deal with the development of an application for the detection of vulnerabilities and their integration into a Ciberintelligence platform, using open source tools and technologies.

Keywords: *Computer Security, Computer Attack, Vulnerability Detection, Software Development, Windows Operating Systems.*

Agradecimentos

Inevitavelmente, as minhas primeiras palavras de agradecimento têm de ir, obrigatoriamente, para os meus pais. Sem o apoio, incentivo, paciência e compreensão, não seria possível concluir mais uma etapa na minha vida académica.

Aos pais da minha namorada, por todo o apoio demonstrado durante os últimos dois anos.

À Susana, minha namorada, por toda a ajuda, dedicação, amor incondicional, muita paciência e principalmente, por abdicar de algum do seu tempo para se dedicar de corpo e alma a ajudar-me em terminar mais uma fase na minha vida.

Agradeço ao Professor Dr. Rui Miguel Silva por toda a dedicação, empenho e disponibilidade que me concedeu ao longo do estágio realizado e pela oportunidade de desenvolver o mesmo na Sparkint.

A todos os meus colegas e professores do Mestrado, por todo o companheirismo, disponibilidade e informações partilhadas, tornando este percurso mais fácil e enriquecedor. Ao João Amarante, por todo o apoio, disponibilidade, dedicação e amizade ao longo destes dois anos de Mestrado.

Ao Mário Candeias por todas as informações e ajuda prestada na fase inicial do estágio. Ao Pedro Moreira, por toda a ajuda e paciência ao longo do estágio, fortalecendo ainda mais os laços de amizade criados durante a Licenciatura. À Neuza Figueira, companheira de luta na fase final do estágio e elaboração do relatório.

Ao João Casaca, pela disponibilidade do espaço para continuar a desenvolver os trabalhos do estágio no final de cada de dia, enquanto esperava pela Susana.

Por último, mas não menos importante, aos meus amigos por compreenderem e aceitarem a minha indisponibilidade durante os últimos tempos, dando sempre o seu contributo para terminar mais uma fase.

Índice

Índice de Figuras

Abreviaturas e Siglas

CCE	Common Configuration Enumeration
CIS	Center for Internet Security
CPE	Common Plataform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
FIRST	Forum of Incident Response and Security Teams
GPL	General Public License
IDE	Integrated Development Environment
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OVAL	Open Vulnerability and Assessment Language
PDF	Portable Document Format
RPC	Remote Procedure Call
SCAP	Security Content Automation Protocol
SQL	Structured Query Language
USGCB	United States Government Configuration Baseline
XML	eXtensible Markup Language

Capítulo 1

Introdução

> "If you spend more on coffee than on IT security, you will be hacked. What's more, you deserve to be hacked."
>
> *Richard A. Clarke - National Coordinator for Security, USA*

Nos dias de hoje, a informação apresenta um carácter intangível no seio de uma organização. A utilização maciça de recursos de processamento e armazenamento da informação através de meios informáticos, nas mais diversas áreas organizacionais, criou uma dependência das infraestruturas informáticas. É difícil imaginar os processos organizacionais sem o uso destas infraestruturas, por exemplo, em empresas de telecomunicações ou em entidades bancárias. Contudo, o impacto gerado pela indisponibilidade dos equipamentos informáticos, não é tão severo como a perda, indisponibilidade ou violação da informação que estes meios suportam. Os crimes informáticos acontecem diariamente, com mais frequência do que em anos anteriores, em todo o mundo com objetivos diversos, colocando frequentemente em risco a informação sigilosa de uma organização.

A segurança informática é um tema bastante presente e relevante para as organizações e para os utilizadores que usufruem diariamente de uma utilização à Internet. A ocorrência de fugas de informação, exposição de informações sensíveis ou a possibilidade de tornar um sistema informático inoperacional é cada vez maior, fomentando sérios constrangimentos numa organização. Estas ocorrências surgem através do aproveitamento de vulnerabilidades. Uma vulnerabilidade é uma falha que surge num determinado sistema ou aplicação, em configurações incorretas ou no

incumprimento de regras básicas aplicadas à segurança da informação.

O aparecimento de vulnerabilidades associadas a aplicações ou sistemas operativos, aumenta significativamente de dia para dia, expondo cada vez mais as organizações e as pessoas que utilizam meios tecnológicos a ataques informáticos.

Para combater este tipo de ameaças informáticas, existem aplicações cujo principal objetivo é a deteção de falhas em sistemas informáticos. Estas aplicações, designadas por *scanners* de vulnerabilidades, verificam se um sistema está ou não a executar um serviço ou aplicação com falhas. Através de uma lista de vulnerabilidades conhecidas, realizam testes de conformidade para verificar o estado atual do sistema, detetando a presença de aplicações, serviços ou configurações passiveis de conterem vulnerabilidades associadas.

Muitas das ferramentas existentes para deteção de vulnerabilidades recorrem a uma linguagem que permite realizar testes de conformidade a sistemas informáticos para a deteção de vulnerabilidades. Os ficheiros utilizados por esta linguagem apresentam informações relevantes sobre estados vulneráveis e aplicações afetadas por vulnerabilidades conhecidas.

Contudo, devido ao grande aumento de vulnerabilidades existentes e em constante crescimento, existe uma grande necessidade de classificar e enumerar cada uma delas, de forma a compreender e apresentar possíveis correções permitindo assim melhorar a avaliação dos riscos que cada uma delas apresenta para uma organização ou individuo. Existem, atualmente, um conjunto de linguagens de classificação e enumeração para lidar com este problema. Podemos dividir estas linguagens em dois sistemas: uma para a gestão de vulnerabilidades e outro para a gestão de configurações. No sistema de gestão de vulnerabilidades, englobam-se as linguagens relacionadas com vulnerabilidades e fraquezas no *software* ou sistema operativo. O outro sistema, representa as linguagens de configurações recomendadas e a lista de produtos (*software* e *hardware*) presentes num sistema.

1.1 Instituição de acolhimento

A **Sparkint – Security & Ciberintelligence Technologies, Lda.**[1] (doravante designada Sparkint), é uma empresa jovem que resultou de um processo de amadurecimento de uma base idealizada por um conceituado *expert* na área da segurança das TI, conjugando o saber académico com a prática e com a realização concreta de projetos. A empresa está localizada no edifício da Escola Superior de Tecnologia e Gestão e Gestão do Instituto Politécnico de Beja.

Os focos do negócio da Sparkint são as avaliações de segurança, conjugadas com o desenvolvimento de produtos específicos. A atividade da empresa é dirigida ao sector empresarial, estatal, privado e da banca. Realiza diversos projetos, desde testes simples a sistemas e aplicações de pequena dimensão, como análises de segurança massivas, contra sistemas e aplicações múltiplas. Além destes projetos, desenvolve atividades especificas na área da consultoria.

A Sparkint é composta por especialistas na avaliação da segurança das soluções críticas de TI que pretendem controlar o nível de risco. Enquanto desenvolve produtos específicos, também fornece proteção acrescida contra ameaças na área da segurança da informação e pugna pela tranquilidade necessária para que o utilizador use as soluções de TI sem problemas.

Os principais serviços prestados pela empresa são:

- Testes de Penetração;
- Teste de Capacidade de Resposta a Incidentes;
- Teste de Espionagem Corporativa;
- Elaboração de Políticas de Segurança da Informação;
- Formação.

Além destes serviços, foi desenvolvida uma plataforma de *Ciberintelligence*, que reúne um conjunto de ferramentas, permitindo aos utilizadores, recolher informações sobre cibersegurança (recolhidas a partir de um conjunto pré-selecionado de fontes abertas), realizar *scans* anónimos para uma obter uma melhor perceção dos riscos que uma determinada organização corre, e trocar mensagens, de forma segura

[1] www.sparkint.pt

e confidencial, entre utilizadores (as mensagens ficam armazenadas nos servidores da Sparkint, cifradas e assinadas digitalmente, garantindo a sua confidencialidade e autenticidade). Através desta plataforma é possível manter os clientes permanentemente informados sobre vulnerabilidades conhecidas, que poderão afetar os seus sistemas, facilitando as tomadas de decisões relacionadas com as medidas necessárias para mitigar potenciais riscos.

A Sparkint procura manter um nível profundo de conhecimento técnico sobre uma vasta gama de temas, relacionados com a segurança, canalizando todo o *know-how* adquirido nesta procura, para proceder ao desenvolvimento de novas soluções de segurança e melhorar os serviços prestados de forma a reduzir os riscos associados a falhas de segurança.

1.2 Âmbito do Estágio

Este relatório de Dissertação de Mestrado, pretende descrever todos os trabalhos desenvolvidos, decisões tomadas e conhecimentos adquiridos durante o período de estágio, realizado na instituição *Sparkint – Security & Ciberintelligence Techonologies, Lda.*, para a obtenção do grau de Mestre, conferido pelo Instituto Politécnico de Beja.

O presente estágio foi realizado na instituição supracitada, durante o período de 2 de maio a 5 de novembro de 2016.

A orientação pedagógica deste estágio esteve a cargo do Professor Dr. Rui Miguel Soares Silva, que desempenhou as funções de orientador da dissertação de Mestrado e na instituição de acolhimento.

O objetivo proposto para a realização deste estágio, consistiu no desenvolvimento de uma aplicação para a deteção de vulnerabilidades em aplicações instaladas em sistemas operativos da família *Windows* integrada na plataforma de *Ciberintelligence* desenvolvida pela Sparkint. A empresa pretende incluir no seu portfólio uma aplicação deste tipo, uma vez que o número de vulnerabilidades conhecidas aumenta todos os dias, aumentando assim a exposição das organizações aos riscos associados à segurança da informação.

Para atingir o objetivo proposto, realizou-se uma análise crítica comparativa das propostas de linguagens de classificação e enumeração de segurança existentes para posteriormente selecionar um conjunto, coerente e com o máximo de complementaridade e completude, dessas mesmas linguagens, de forma a serem aplicadas no desenvolvimento da aplicação.

Após esta análise, foram estudadas ferramentas, plataformas e projetos em desenvolvimento, comerciais ou de código aberto, proprietários ou normalizados, que implementam uma ou várias das linguagens selecionadas.

O desenvolvimento da aplicação teve como base a arquitetura cliente/servidor, permitindo assim aos utilizadores, realizar o download do cliente na plataforma da Sparkint e analisar os resultados obtidos no mesmo local. Para a elaboração deste projeto, ficou assente desde uma fase muito prematura que, todas as tecnologias utilizadas no desenvolvimento da aplicação deveriam ser de código aberto ou gratuitas.

1.3 Organização do documento

O presente relatório de estágio encontra-se organizado em seis capítulos, sendo eles: Introdução, Fundamentação Teórica, Solução Proposta, Implementação da Aplicação, Avaliação e Conclusão e Trabalho Futuro. Esta estrutura pretende abordar de uma forma organizada e concreta o trabalho desenvolvido ao longo do estágio, permitindo obter uma ideia clara e concisa, desde a proposta de estágio, passando pela fundamentação para as decisões tomadas, o desenvolvimento da aplicação, e finalizando com um conjunto de conclusões retiradas.

No Capítulo 1, **Introdução**, é realizada a introdução ao presente relatório de estágio, onde é mencionado o contexto académico, assim como a descrição alargada da instituição de acolhimento e o âmbito do estágio.

O Capítulo 2, **Fundamentação Teórica**, visa apresentar fundamentos, conceitos e um conjunto de aplicações relacionadas, com o objetivo do estágio, para uma melhor compreensão dos trabalhos realizados.

No Capítulo 3, **Solução Proposta**, encontra-se descrito detalhadamente, a solução inicialmente proposta e a versão final que foi desenvolvida durante este período de seis meses.

O Capítulo 4, **Implementação da Aplicação**, descreve as tecnologias e ferramentas utilizadas, a estrutura do repositório de ficheiros e o processo de desenvolvimento das duas partes que compõe a aplicação.

No Capítulo 5, **Avaliação**, são abordados os testes realizados para testar o funcionamento da aplicação, a metodologia utilizada para os mesmos e a avaliação geral da aplicação.

Por fim, no Capítulo 6, **Conclusão e Trabalho Futuro**, realiza-se uma reflexão conclusiva sobre o trabalho desenvolvido e os resultados obtidos e sugerem-se várias melhorias a serem desenvolvidas num trabalho futuro.

Capítulo 2

Fundamentação Teórica

Com o avançar da tecnologia e com o rápido aumento do número de utilizadores, as organizações abrem, cada vez mais, os seus Sistemas de Informação (SI) aos seus parceiros, clientes e fornecedores, aumentando assim a exposição das mesmas ao mundo exterior. Com isto, existe um claro aumento das preocupações a nível de segurança da informação. Existem três grandes razões que as empresas referem para justificar estas preocupações:

- **Dependência** - sem sistemas informáticos, grande parte das organizações ficariam inoperacionais ou bastante limitadas ao seu bom funcionamento, uma vez que o fornecimento de serviços, processamento de documentos e contactos dependem dos SI;
- **Vulnerabilidade** - sendo estes sistemas bastante importantes para o armazenamento e acesso a uma vasta quantidade de dados organizacionais sigilosos, tornam-se um alvo apetecível para os "ciber criminosos" (comummente designados de Hackers e Crackers [1]) . A divulgação destas informações pode causar grandes transtornos.
- **Investimento** - por norma, os SI, são sistemas caros, no que respeita ao desenvolvimento e manutenção. A administração de uma organização deve proteger esse investimento, tanto a nível físico como a nível lógico, dando a mesma importância como a qualquer outro ativo considerado valioso para a organização.

2.1 Segurança da Informação

A necessidade de combater ataques informáticos é cada vez mais importante para as organizações e também para os utilizadores que usufruem da Internet. Qualquer computador ligado em rede é um potencial alvo para sofrer ataques informáticos. Software malicioso, como vírus ou worms, e ataques em rede, são cada vez mais sofisticados tornando-se cada vez mais difícil a mitigação de potenciais vulnerabilidades. Erros de programação, utilizadores incautos ou não sensibilizados para os problemas da segurança da informação e inexistência de mecanismos de defesa são alguns exemplos que dificultam a mitigação destes ataques e motivam a propagação de novos. Qualquer pessoa pode colocar uma página web com código malicioso na rede tornando-se desde logo acessível a um número ilimitado de utilizadores. Novas realidades, novas vulnerabilidades e o desenvolvimento tecnológico, oferecem um conjunto de oportunidades para que os atacantes consigam que as suas ações atinjam os objetivos desejados.

A segurança da informação procura reduzir os riscos de escoamento, fraude, erros, uso indevido, roubo de informações ou qualquer outra ameaça que possa prejudicar os sistemas de informação ou equipamentos de um individuo ou organização.

Tal como define a ISO/IEC 27002[1], "***Information security*** *is explicitly defined as the "preservation of confidentiality, integrity and availability of information*", um sistema ou aplicação é dito como seguro se atender aos três pilares:

- **Confidencialidade** – refere-se à proteção de dados e informações trocadas entre um emissor e um ou mais destinatários, contra terceiros. Isto deve ser feito independentemente da segurança do sistema de comunicação utilizado: de fato, uma questão de grande interesse é o problema de garantir o sigilo de comunicação utilizado quando o sistema é à partida inseguro (como por exemplo, o uso da Internet).

- **Integridade** – significa ter disponível informações confiáveis, corretas e dispostas em formato compatível com o de utilização, ou seja, informações íntegras. Significa que a informação não foi alterada de forma não autorizada ou indevida. Se a informação é alterada de forma errada ou mesmo falsificada, ela

[1]Código de Boas Práticas para a Gestão da Segurança da Informação. Criado por duas organizações internacionais que padronizam tecnologias, a ISO (International Organization for Standardization) e a IEC (International Electrotechnical Commission).

perde a sua eficácia e credibilidade, tornando vulneráveis as decisões que a partir dela são tomadas, e tirando a credibilidade do ambiente que a forneceu.

- **Disponibilidade** – garante que uma informação estará disponível para acesso no momento desejado. Assegura ao utilizador o acesso à informação sempre que este precisar da informação.

Através da correta aplicação destes princípios, a segurança da informação pode trazer benefícios para as organizações, tais como, aumentar a produtividade, maior controlo sobre os recursos informáticos, redução de incidentes e maior credibilidade a nível de proteção da informação.

2.1.1 Importância da Segurança

A informação é algo de grande importância para uma organização e fundamental para os negócios. Para isso, é importante que seja protegida de forma adequada. Com o crescimento da ligação entre ambientes de trabalho, a informação fica exposta a uma grande variedade de ameaças. Daí a necessidade da segurança da informação, que é a proteção da informação contra os vários tipos de ameaças, minimizando os riscos relacionados com o negócio, e maximizando o retorno sobre os investimentos.

Para obter a segurança da informação, torna-se necessário um conjunto de controlos e procedimentos adequados, com o intuito de garantir que os objetivos do negócio e de segurança da organização sejam conseguidos. Esses controlos têm vindo a ser alterados e aperfeiçoados com o passar do tempo, permitindo que as organizações cuidem e se previnam contra eventuais riscos causados pela falta de segurança.

Para se ter um ambiente tecnológico seguro, é necessário implementar um conjunto de normas e procedimentos para a gestão da segurança da informação, que devem ser do conhecimento de todos os elementos da organização. Além da implementação, deverá ser realizado um esforço por parte da organização para manter e atualizar tais normas e procedimentos de acordo com a evolução das tecnologias, mudanças na organização e novas ameaças à segurança da informação.

Mas porque nos devemos preocupar com a segurança da informação?

Simplesmente porque este tema se tornou importante na sociedade contemporânea, visto que as empresas guardam nos computadores os segredos de seus

negócios. Por outro lado, as pessoas cada vez mais expõem dados de carácter pessoal ao utilizarem os meios tecnológicos.

Todos têm o legítimo direito de esperar que os dados confiados às máquinas sejam mantidos intactos e confidenciais, acessíveis apenas às pessoas autorizadas. Apesar de todas as ferramentas de hardware e de software disponíveis no mercado, há sempre uma possibilidade de falhas de segurança. Alguns especialistas defendem mesmo a ideia de que é impossível ter um ambiente tecnológico totalmente seguro.

Segundo Kevin Mitnick[2], "*não existe uma rede segura; nem um computador seguro. O único computador seguro é aquele que está desligado da tomada, fechado numa caixa forte, e a única pessoa que conhecia a combinação morreu na semana passada*". Através da análise destas palavras, podemos observar que a segurança da informação é de uma importância elevada pois é a informação que está em risco e a informação é o ativo mais valioso para uma organização.

2.2 Vulnerabilidades

Segundo RFC2828[2], vulnerabilidade "é uma falha no sistema que pode ocorrer no projeto, na implementação, na operação ou gestão deste e que pode ser explorada para violar a sua política de segurança". As vulnerabilidades dos SI são uma porta aberta para os "ciber criminosos" acederem à informação que estes sustentam. É vital identificar essas vulnerabilidades e corrigi-las de forma a evitar ataques informáticos. Estas falhas podem ser provenientes de:

- **Bugs** - os *bugs* surgem a partir do momento em que o programador de um determinado *software* comete um erro. Estão presentes em qualquer programa, desde um simples editor de texto até ao próprio sistema operativo. Podem provocar falhas na segurança geral do computador ou da rede;

- **Configurações de segurança incorretas** - os administradores de uma organização, necessitam contratar especialistas com conhecimentos suficientes para detetar e corrigir as falhas de um sistema. Caso contrário, acabarão por surgir várias falhas no sistema como, ficheiros sem proteção, vulnerabilidades em mecanismos de controlo de acesso, execução de código malicioso, etc;

[2]Kevin David Mitnick é um programador e hacker, conhecido mundialmente a partir da década de 1990. Atualmente trabalha como gerente de uma empresa de segurança.

- **Desinteresse dos Administradores pela Segurança** - falta de políticas de segurança ou planos de contingência. Muitas vezes nem sequer existe Antivírus nas máquinas da organização e os *updates* de segurança estão totalmente desatualizados ou são inexistentes;

- **Incumprimento das regras básicas de segurança** - Os utilizadores são constantemente considerados o "elo mais fraco"do sistema, gerando com as suas ações, várias vulnerabilidades, tais como, a utilização de passwords intuitivas e fáceis de decifrar, utilização ou acesso a serviços duvidosos e inseguros, falta de cuidado com a navegação na Internet e abertura de ficheiros desconhecidos.

2.3 Classificação de Vulnerabilidades

Nos dias de hoje, não existe ainda um consenso sobre a forma correta de classificação de vulnerabilidades, contudo, existem diversos grupos e organizações de pesquisa que mantêm classificações diferentes entre si, mas que apresentam uma possível forma de interligação. Dentro destas classificações é possível destacar as seguintes: CWE, CVE, CVSS, CPE e CCE. Além de existirem estes tipos de classificação, é também importante destacar a existência de uma linguagem para realizar testes de conformidade (OVAL) e uma plataforma para a automatização de informações de segurança (SCAP).

2.3.1 CWE - Common Weakness Enumeration

A **CWE** [3] é uma lista (ou dicionário) de fraquezas que podem ocorrer na arquitetura, desenho, código ou na implementação do software. É mantida pela MITRE Corporation e encontra-se disponível de forma gratuita para qualquer organização ou individuo que pretenda utilizá-la para pesquisa, desenvolvimento e/ou fins comerciais.

Esta linguagem foi criada com o objetivo de descrever falhas de segurança do software, direcionar as ferramentas de segurança contra este tipo de fraquezas e servir de linha mestra para a identificação, mitigação e prevenção destas fraquezas.

2.3.2 CVE - Common Vulnerability Enumeration

A **CVE** [4] é uma lista de identificadores padronizados para vulnerabilidades de cibersegurança publicamente conhecidos. Esta linguagem também é mantida pela MITRE Corporation. Através da utilização de identificadores é mais fácil partilhar informações sobre vulnerabilidades.

A classificação CVE atribui um identificador único a cada vulnerabilidade encontrada, sendo este composto pelo ano em que a vulnerabilidade foi descoberta, seguido de um hífen e posteriormente um valor de quatro dígitos (exemplo: CVE-2016-0069). Após aprovação pelo conselho responsável (CVE Board[3]), a nova vulnerabilidade é adicionada à lista de CVE's.

2.3.3 CVSS - Common Vulnerability Scoring System

A classificação **CVSS** [5] (atualmente na versão 3.0 [6]) consiste num sistema de atribuição de uma pontuação de severidade a cada vulnerabilidade conhecida e foi estruturada com o objetivo de adotar um método padronizado e aberto de classificação de vulnerabilidades na área da cibersegurança.

Mantida pela FIRST (Forum of Incident Response and Security Teams), esta classificação permite às organizações implementar e coordenar respostas mais eficazes na mitigação de vulnerabilidades de segurança.

A pontuação atribuída às vulnerabilidades conhecidas, é calculada com recurso a métricas distintas, admitindo valores entre 0 (zero) e 10 (dez), sendo que 10 é o valor atribuído a uma vulnerabilidade considerada grave.

As métricas utilizadas por esta classificação são:

- Métricas Base - capturam essencialmente as características da vulnerabilidade que se mostram constantes ao longo do tempo e através de ambientes do utilizador;
- Métricas Temporais – procuram avaliar a evolução das vulnerabilidades ao longo do tempo.
- Métricas Ambientais – fornecem informações sobre a importância dos sistemas afetados para as organizações e para os seus funcionários ou clientes [7].

[3]https://cve.mitre.org/community/board/

2.3.4 CPE - Common Platform Enumeration

A linguagem **CPE** [8] consiste num método padronizado para descrever e identificar classes de aplicações, sistemas operativos e hardware presentes nos ativos de uma determinada organização. Esta linguagem pode ser utilizada como, uma fonte de informação sobre vulnerabilidades ou configurações ou para reforçar e verificar a gestão das políticas de segurança da informação relacionadas com esses ativos.

A linguagem CPE (versão 2.3) foi lançada pelo NIST (National Institute of Standards and Technology [9]) em agosto de 2011 e continua a ser mantida por esta organização. Esta versão substitui a versão 2.2, que foi lançado em março de 2009. Esta nova versão afasta-se significativamente da prática utilizada na versão 2.2 uma vez que, quebra o padrão de classificação CPE num conjunto de especificações, separadas e organizadas numa pilha, contendo Naming, Name Matching, Dictionary e Language. O Naming é a base da pilha e cada especificação é construída sobre aquelas que a precedem.

2.3.5 CCE - Common Configuration Enumeration

A linguagem **CCE** [10] é uma lista de identificadores padronizados para configurações de segurança publicamente conhecidas. Tal como a linguagem CPE, é mantida pelo NIST, após transitar da MITRE Corporation[11]. Através da utilização de identificadores, é mais fácil partilhar informações sobre configurações.

Tal como acontece na linguagem CVE, a classificação CCE atribui um identificador único a um determinado problema de configuração relacionado com segurança. Estes identificadores estão associados a instruções de configuração que expressam a maneira como devem ser configurados sistemas e dispositivos informáticos. Com a utilização de identificadores, é estabelecida uma ponte entre a linguagem natural e as ferramentas desenvolvidas através de sistemas informáticos (como por exemplo, ferramentas de auditoria a configurações).

Cada entrada na lista de CCE contém os seguintes cinco atributos:

- Número de Identificação - "CCE-2715-1";
- Descrição - uma descrição humanamente compreensível do problema de configuração;
- Parâmetros conceituais - parâmetros específicos para implementar um CCE num sistema;
- Mecanismos Técnicos Associados - para qualquer problema de configuração pode haver uma ou mais maneiras de implementar o resultado desejado;
- Referências - referências para as secções específicas dos documentos ou ferramentas em que o problema de configuração é descrito em detalhe.

2.3.6 OVAL - Open Vulnerability and Assessment Language

A linguagem OVAL [12] surge de um esforço da comunidade internacional para criar um padrão de avaliação e descrição do estado de um determinado ativo (computador, servidor, etc.). O OVAL inclui uma linguagem para codificar detalhes do sistema e um vasto conjunto de repositórios com conteúdos, mantidos pela comunidade.

Esta linguagem (Figura 2.1) fornece uma estrutura para realizar testes relativos ao estado de uma determinada máquina, através de três processos padronizados:

- Representação do estado do sistema para a realização de testes;
- Análise do sistema para a verificação do estado da mesma em relação a vulnerabilidades, configurações, patch, etc.;
- Descrição dos resultados da análise.

Os ficheiros utilizados pela linguagem, como os ficheiros sobre definições de vulnerabilidades, encontram-se disponíveis em vários repositórios distribuídos por toda a comunidade que contribui para o desenvolvimento da linguagem. Um desses repositórios, o repositório hospedado pela CIS – Center for Internet Security (até 2015 era da responsabilidade da MITRE Corporation [13]), é o ponto de encontro central para a comunidade OVAL discutir, analisar, armazenar e disseminar os ficheiros que compõem a linguagem.

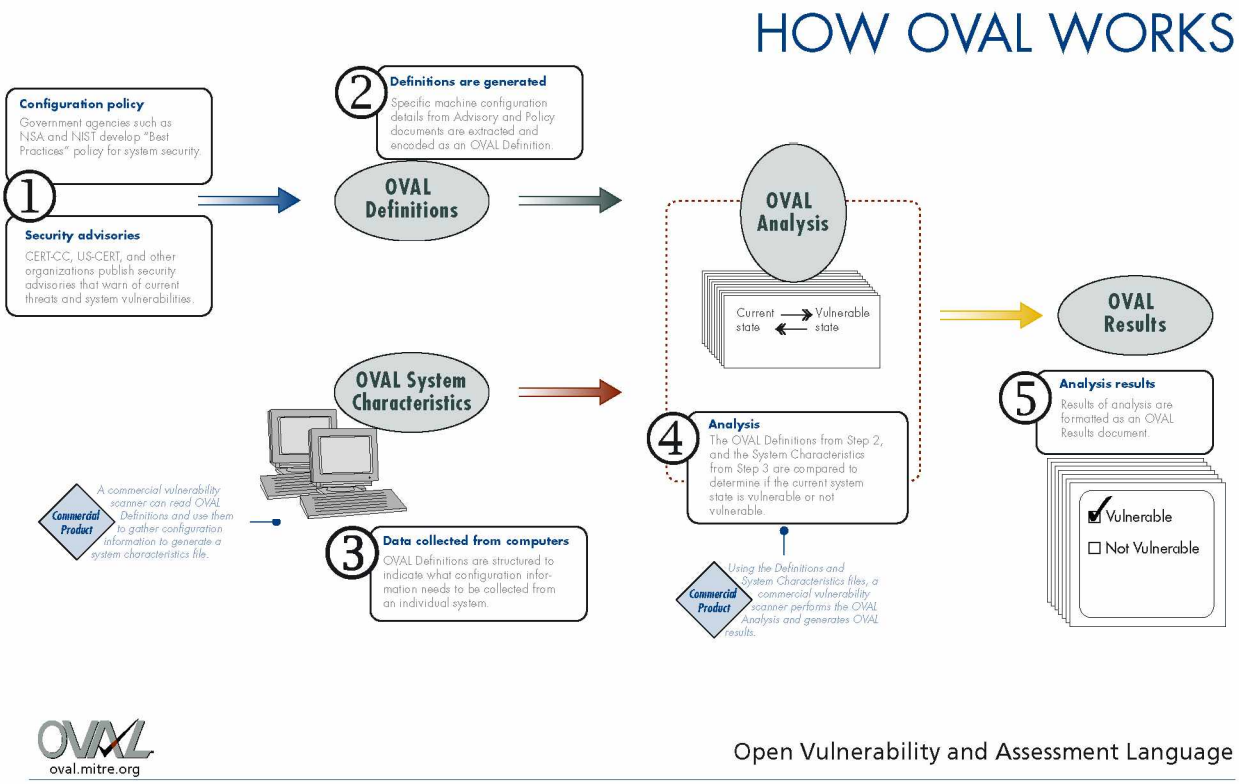


Figura 2.1: Funcionamento do OVAL

2.3.7 SCAP - Security Content Automation Protocol

Para aumentar a proteção contra ameaças de cibersegurança, as organizações precisam de monitorizar de forma continua os sistemas e as aplicações de computador que implementam, incorporar atualizações de segurança no software e realizar atualizações nas configurações. O **SCAP** [14] compreende uma série de padrões abertos, amplamente utilizados, para enumerar falhas de software e problemas de configurações relacionados com a segurança. As aplicações que realizam uma monitorização de segurança, utilizam padrões que analisam os sistemas para detetar vulnerabilidades e oferecem métodos para classificar essas mesmas vulnerabilidades, a fim de avaliar o possível impacto na segurança das organizações.

Uma ferramenta de deteção de vulnerabilidades e configurações baseada no protocolo SCAP, compara as configurações e estados de uma determinada aplicação ou computador com a linha base presente no SCAP [15]. Após esta análise, a aplicação gera um relatório com as informações encontradas. Algumas ferramentas conseguem

corrigir as falhas encontradas de forma a ficarem em conformidade com a linha base estabelecida pelo SCAP.

O SCAP é mantido pelo NIST e recorre às ferramentas disponíveis na NVD (National Vulnerability Database [16]).

2.4 Panorâmica sobre sistemas de classificação

Tal como proposto por Silva [17], é possível realizar uma divisão, em dois tipos, de sistemas de classificação: um para a gestão de vulnerabilidades e outro para a gestão de configurações.

O sistema de classificação para a gestão de vulnerabilidades, tal como mostra a Figura 2.2, engloba as listas de vulnerabilidades conhecidas (CVE), listas de fraquezas no software (CWE), sistema de classificação de vulnerabilidades (CVSS), linguagem para testes de conformidade (OVAL) e uma plataforma de normalização de relatórios sobre vulnerabilidades.

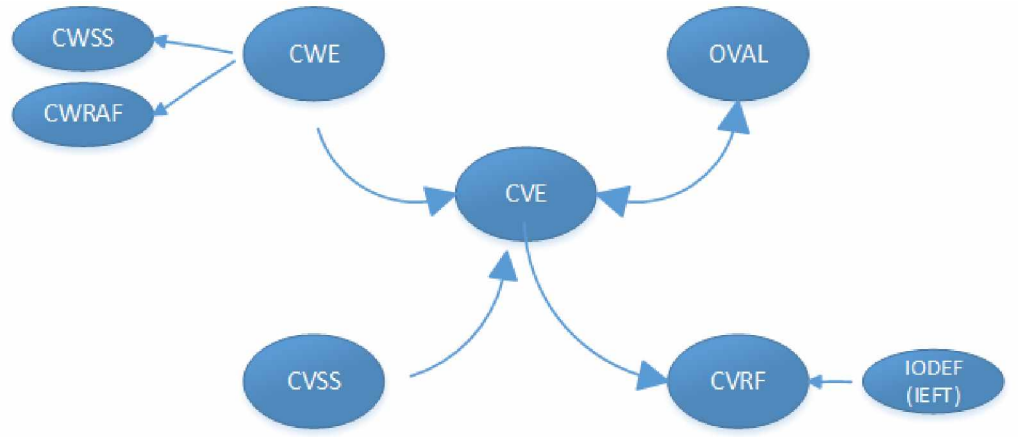


Figura 2.2: Sistema de Classificação para a gestão de Vulnerabilidades

Já o sistema de classificação para a gestão de configurações (Figura 2.3), inclui as configurações recomendadas (CCE), a lista de produtos (CPE), uma plataforma de automatização de informações de segurança (SCAP)e também a linguagem para testes de conformidade (OVAL).

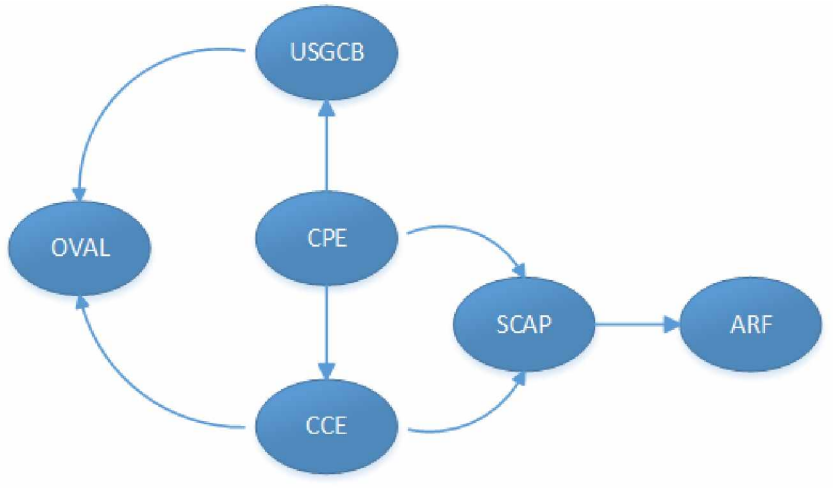


Figura 2.3: Sistema de Classificação para a gestão de Configurações

2.5 Metodologias de Ataques Informáticos

Um ataque informático consiste em aproveitar uma debilidade ou falha (vulnerabilidade) no software, no hardware, e em alguns casos, nas pessoas que fazem parte de um ambiente informático. Com o fim de obter algum beneficio, normalmente económico, os ataques informáticos afetam a segurança dos sistemas, com consequência direta nos ativos de uma organização. Estes ataques podem ser implementados segundo várias metodologias ou técnicas:

Vírus Informático - Segmento de código malicioso, normalmente anexo a outros programas, desenvolvido para se copiar a si próprio, de computador em computador, de forma a espalhar-se rapidamente pelos recursos do sistema, propagando a infeção à medida que circula. Os vírus informáticos, geralmente, atacam programas e o sector de arranque do disco, destroem ficheiros, modificam dados e comprometem sistemas.

Worms - são uma subclasse de vírus informático. Porém, ao contrário deste último, espalham-se sem a interação do utilizador, distribuindo cópias de si próprio através da rede e tomando o controlo de funções do computador que permitem transportar informação. Depois de entrar no sistema, um *worm* movimenta-se sozinho e multiplica-se em grande volume, podendo, para além de outros tipos de danos, consumir memória ou largura de banda, fazendo com que o computador fique bloqueado.

Trojans ou Cavalos de Troia - são programas disfarçados, aparentando ser *software* útil, que executam uma determinada tarefa maligna, com o intuito de comprometer a segurança do sistema. Acontece com bastante frequência um utilizador executar um jogo adquirido através da Internet, que secretamente, instala um Trojan que abre uma porta TCP para possibilitar uma invasão ao sistema.

Spyware - estes programas monitorizam e reportam o uso que fazemos dos programas, especialmente o uso da Internet, transmitindo toda a informação do utilizador para uma entidade externa sem o conhecimento ou consentimento deste. São utilizados, sobretudo, como ferramentas de marketing, e muitas vezes estão na origem de correio eletrónico de *spam*.

2.6 Scanner de Vulnerabilidades

Um *scanner* de vulnerabilidades tem como objetivo avaliar uma variedade de falhas em sistemas de informação (incluindo computadores, sistemas de redes, sistemas operativos e aplicações) que podem ter sido originadas por um vendedor/empresa de *software*, atividades relacionadas com a administração de sistemas ou atividades comuns realizadas no dia-a-dia pelos utilizadores:

- **Vendedor/empresa de software** – inclui bugs no software, ausência de correções ao sistema operativo, serviços vulneráveis, configurações inseguras e aplicações web vulneráveis;
- **Administração de sistemas** – integra alterações incorretas ou não autorizadas às configurações do sistema, falta de políticas de segurança da informação (gestão de passwords, backups, acessos físicos, etc.);
- **Utilizador** – abrange a partilha de pastas/ficheiros com partes não autorizadas, falta de antivírus no computador ou outro tipo de *software* de deteção de *software* malicioso, utilização de aplicações que pode conter código malicioso.

Um *scanner* de vulnerabilidades deve apresentar os seguintes requisitos:

- Uma base de dados atualizada de vulnerabilidades;
- Evitar falsos positivos ou falsos negativos nos resultados da análise;
- Fornecer informações relevantes sobre os problemas encontrados de forma a permitir as correções necessárias.

Através de uma lista de falhas conhecidas, o *scanner* verifica se o sistema está ou não a executar um serviço com problemas. O *scanner* deve ser capaz de detetar erros comuns de configuração (como por exemplo, *software* com configurações de fábrica) e vulnerabilidades conhecidas. Existem várias aplicações no mercado que executam análises de vulnerabilidades de uma rede ou máquina.

2.7 Aplicações Relacionadas

Foram selecionadas quatro aplicações que apresentam características relacionadas com a solução proposta para a realização deste estágio. A seguir são detalhadas as ferramentas: OpenVAS, LanGuard, Nessus e QualysGuard.

2.7.1 LanGuard

Languard [18] foi criado em 2000 pela empresa GFI Software, é um scanner de segurança, que oferece uma solução de gestão de atualizações de segurança e auditoria de rede. Tem como objetivo assegura o bom funcionamento de uma rede local. Este software oferece uma gestão de atualizações de segurança para todas as famílias de sistemas operativos e também para aplicações. Possui a capacidade de detetar vulnerabilidades de rede antes de serem expostas e realizando a avaliação para detetar ameaças de seguranças através da avaliação de um grande número de vulnerabilidades conhecidas. Esta avaliação inclui sistemas de virtualização e equipamentos de rede, tais como *switchs*, *routers*, pontos de acesso e impressoras. Através da auditoria de rede é possível analisar o estado da segurança da rede, identificar os riscos e o grau de exposição da mesma. Oferece uma visão completa das aplicações instaladas, hardware presente na rede, dispositivos móveis e o estado de segurança das aplicações, como antivírus, *anti-spam*, *firewall*, entre outros. A Figura 2.4 mostra o funcionamento da aplicação Languard.

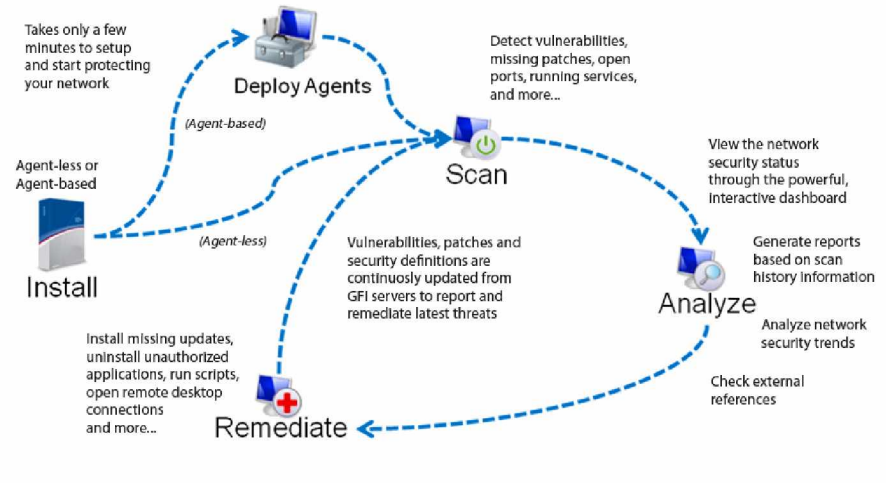


Figura 2.4: Funcionamento da aplicação Languard

2.7.2 Nessus Vulnerability Scanner

O Nessus [19] foi lançado em 1998, sob a licença GPL (*General Public License*). Até à versão 2.2 era uma alternativa *open-source*, mas a partir da versão 3.0, a *Tenable Network Security* decidiu fechar o código-fonte, uma vez que muitas empresas passaram a incorporar recursos do Nessus nos seus produtos e a desenvolver versões modificadas da solução oferecida por eles.

O Nessus Vulnerability Scanner é uma excelente ferramenta usada para identificar vulnerabilidades e falhas na rede local. Esta ferramenta faz parte de uma plataforma desenvolvida pela empresa (ver Figura 2.5). Através de uma deteção de portas, deteta servidores ativos, simulando ataques para detetar vulnerabilidades. É uma ferramenta multi plataforma composta por uma arquitetura cliente/servidor. O servidor é responsável pela análise da rede e deteção de falhas de segurança e o cliente apresenta uma interface gráfica e permite realizar análises de vulnerabilidades.

Além destas funcionalidades, o Nessus permite realizar auditorias remotas e determinar se a rede foi invadida ou usada de maneira indevida, detetar uma vasta gama de dispositivos físicos e virtuais na rede e identificar sistemas operativos, aplicações, bases de dados e serviços em execução nos dispositivos de rede.

Figura 2.5: Estrutura da plataforma da Tenable Network Security

2.7.3 Qualys Vulnerability Management

A Qualys foi fundada em 1999 e a empresa lançou o QualysGuard em dezembro de 2000. Esta empresa, tal como a *Tenable Network Security*, foi uma das pioneiras no desenvolvimento de aplicações para a deteção e gestão de vulnerabilidades Qualys Vulnerability Management [20] é um serviço baseado na nuvem, que fornece uma visibilidade imediata e global sobre vulnerabilidades e ameaças que podem afetar os ativos de uma organização e também apresenta formas para proteger os mesmos.

Além de identificar potenciais ameaças, também monitoriza alterações inesperadas na rede que podem levar a eventuais falhas, relatórios centralizados (incluem o uso de referencias à linguagem CVE para facilitar a identificação das vulnerabilidades detetadas) e monitorização de correções.

Integrado numa plataforma de segurança da informação desenvolvida pela Qualys, é uma solução bastante avançada para a gestão de vulnerabilidades e também para a deteção de ativos presentes na rede da organização.

A Figura 2.6 apresenta o ciclo da gestão de vulnerabilidades da ferramenta Qualys Vulnerability Management.

Figura 2.6: Estrutura da ferramenta Qualys Vulnerability Management

2.7.4 OpenVAS

O OpenVAS [21] é um sistema para avaliação de vulnerabilidades de código aberto, distribuído sob a licença GPL, sendo uma derivação livre do software de vulnerabilidades Nessus. O objetivo inicial do projeto era permitir o livre desenvolvimento do atual *Nessus Vulnerability Scanner*. Atualmente, o OpenVAS possui uma comunidade em crescimento que contribui para a melhoria e também documentação do software. Existem versões disponíveis para Linux e Windows.

Esta ferramenta oferece um ambiente completo para a avaliação da segurança, através de uma vasta lista de serviços e componentes que podem ser organizados de diversas formas, permitindo assim construir um ambiente de avaliação adequado à rede em análise. Tal como o Nessus, segue o modelo cliente/servidor. O servidor é o componente central, que contém as funcionalidades utilizadas para a execução de um grande número de testes e é responsável pelo agendamento e execução das deteções. O cliente é constituído pela interface gráfica, onde é possível configurar as atividades, deteções e aceder aos resultados obtidos. A Figura 2.7 mostra o funcionamento da aplicação OpenVAS.

O *scanner* de segurança é atualizado diariamente com testes de vulnerabilidades, denominados *Network Vulnerability Test* (NVT). Os NVT são rotinas de testes que verificam a presença de uma vulnerabilidade num determinado sistema. Estes testes são desenvolvidos na linguagem de *script* do Nessus (NASL).

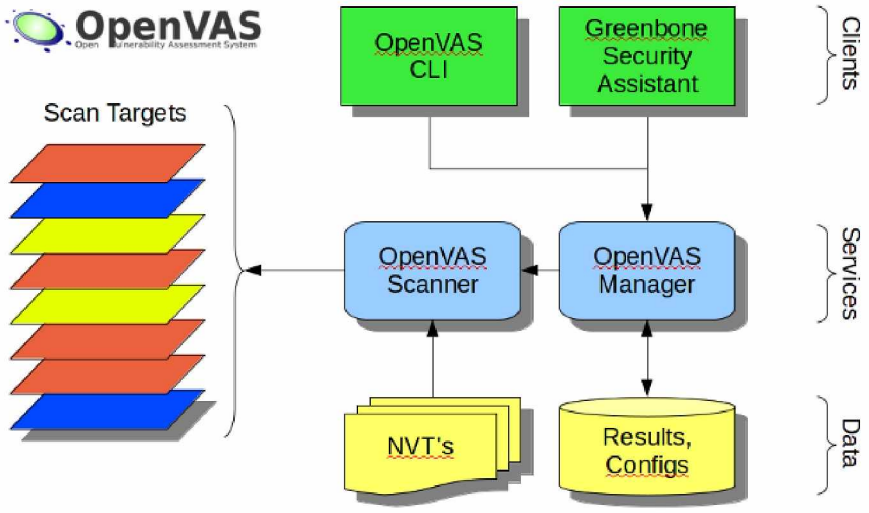


Figura 2.7: Funcionamento da aplicação OpenVAS

Capítulo 3

Solução Proposta

A segurança dos sistemas informáticos é um sistema complexo, dependendo de inúmeros fatores e de naturezas muito diversas e desconexas. Existem várias abordagens estruturadas de classificação, de sistematização de procedimentos e de produção de resultados no domínio da segurança dos sistemas informáticos. No entanto, as ferramentas de trabalho com base nestas abordagens são muitas vezes de custo elevado, ou, sendo de código aberto, os resultados que apresentam não correspondem com precisão a um resultado final útil para quem necessita avaliar a segurança, o risco, ou que procedimentos se devem tomar para corrigir ou melhorar a segurança dos sistemas informáticos.

A Sparkint – Security & Ciberintelligence Technologies, Lda. pretende acrescentar ao seu portfólio uma aplicação que permita realizar a deteção de vulnerabilidades, configurações e fraquezas existentes num determinado conjunto de aplicações instaladas num sistema operativo *Windows*. Após a realização dos testes, a aplicação deve fornecer informações detalhadas sobre os resultados obtidos após a execução da ferramenta.

Para tal, a aplicação deve utilizar uma série de linguagens de classificação e enumeração de segurança, previamente selecionadas, de forma a obter os melhores resultados após a execução da aplicação. Um ponto fundamental para o desenvolvimento desta ferramenta, é a necessidade de implementar todos os componentes inerentes ao desenvolvimento recorrendo apenas a tecnologias *open source*.

3.1 Proposta Inicial

Através de uma longa pesquisa sobre linguagens de classificação e enumeração de segurança, optou-se por utilizar as seguintes linguagens:

- **CPE** - Common Platform Enumeration;
- **CVE** - Common Vulnerability Enumeration;
- **CCE** - Common Configuration Enumeration;
- **CWE** - Common Weakness Enumeration.

Após a análise detalhada dos ficheiros disponíveis com informações sobre vulnerabilidades e software instalado, optou-se por utilizar os ficheiros disponíveis na linguagem OVAL, presentes no repositório oficial[1], para testar vulnerabilidades numa determinada máquina. Foram selecionados dois tipos de ficheiros presentes no repositório:

- **Inventory** – ficheiro com informações de testes a realizar para detetar o software presente numa determinada máquina;
- **Vulnerability** – ficheiro com testes a serem realizados para a identificação de vulnerabilidades.

No que diz respeito à deteção do software instalado, a abordagem adotada distingue-se da utilizada pela maioria das ferramentas que recorrem aos ficheiros da linguagem OVAL, uma vez que a ferramenta desenvolvida primeiro deteta o software instalado e o sistema operativo da máquina a ser testada, e depois é que realiza os testes de vulnerabilidade de acordo com as vulnerabilidades disponíveis para cada software no sistema operativo detetado.

Com esta abordagem pretende-se reduzir o tempo de realização dos testes e o número de testes a realizar pela aplicação, uma vez que a aplicação não irá detetar todo o *software* presente no ficheiro de inventário. Este tipo de abordagem pretende ser uma fator de distinção do restante software para deteção de vulnerabilidades existentes no mercado.

[1] https://oval.cisecurity.org/repository

Relativamente à linguagem de enumeração CCE e CWE, foram selecionados os ficheiros presentes nos sites oficiais de cada linguagem, realizando o download dos dicionários.

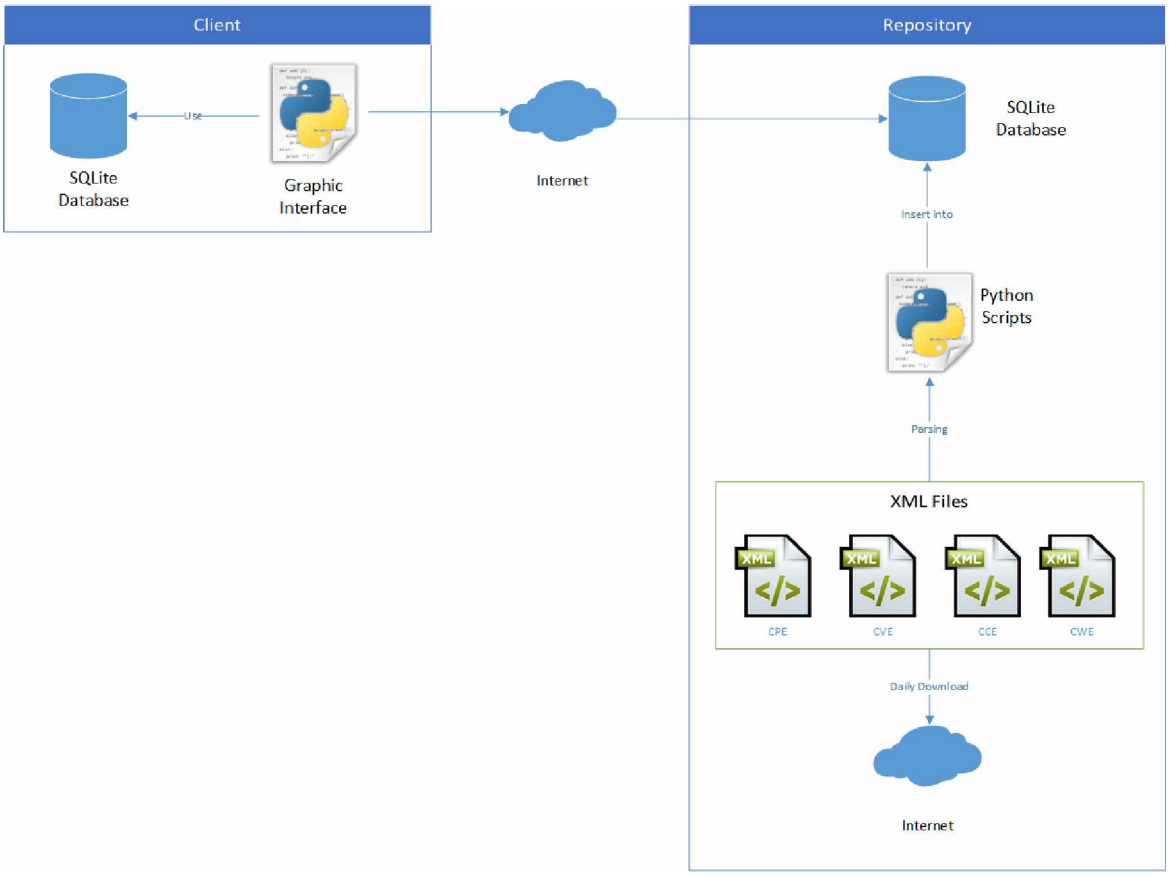


Figura 3.1: Proposta Inicial da Aplicação

A Figura 3.1 mostra a primeira abordagem adotada para implementar esta aplicação, que consistia em dividir a aplicação em duas partes: um cliente com interface gráfica e um repositório (online) para alojar os ficheiros em XML (com as informações sobre vulnerabilidades, configurações, fraquezas e software instalado) e a base de dados (resultante da transformação dos ficheiros descarregados).

No lado do repositório, seriam implementados *scripts* para realizar o download (diariamente e de forma automática) dos ficheiros necessários para criar uma base de dados com toda essa informação agrupada e de fácil acesso de forma a ser utilizada pelo cliente.

Inicialmente a base de dados foi projetada para ser implementada através da biblioteca *SQLite*[2], pois este motor de base de dados é um software livre e multi plataforma, não necessita de instalação, configuração ou administração e também porque permite guardar a base de dados num ficheiro único.

3.1.1 Funcionamento da aplicação

Tal como foi referido anteriormente, a aplicação consiste em duas partes com funções independentes, mas complementares entre si. No lado do repositório serão realizadas as operações de download e criação da base de dados, bem como tarefas de manutenção do mesmo repositório.

Diariamente é realizado o download dos ficheiros necessários para obter todas as informações e caso estes sejam diferentes, a base de dados é atualizada. Esta verificação é realizada comparando os valores obtidos através da função de *hash* com MD5[22] do ficheiro atual e do ficheiro descarregado diariamente. Se estes forem diferentes, o ficheiro é substituído e automaticamente é realizada uma atualização à base de dados com as novas informações.

Em relação ao cliente, a primeira tarefa a ser executada é a verificação da existência de uma base de dados atualizada no repositório, também recorrendo à comparação do *hash* de ambas as bases de dados. Contudo, a aplicação pode funcionar sem ser necessária uma ligação à Internet, pois a base de dados pode ser utilizada *offline*.

Após esta verificação, a aplicação procede à deteção do software instalado, realiza os testes necessários para a deteção de vulnerabilidades, configurações e fraquezas presentes na máquina a ser testada.

No final desta deteção, é gerado um relatório em PDF com todas as informações relacionadas com:

- Aplicações e Sistema operativo - lista das aplicações instaladas e o sistema operativo, devidamente identificadas com o seu CPE (exemplo: cpe/:a:google:chrome);

[2]https://sqlite.org/

- Vulnerabilidades - falhas associadas a uma determinada aplicação ou sistema operativo, identificadas através do ID associado ao standard para deteção de vulnerabilidades e a descrição da mesma (por exemplo, CVE-2015-6768);
- Configurações recomendadas - listas de todas as configurações relacionadas com as aplicações e sistema operativo detetado. Cada configuração é devidamente identificada com o seu ID associado ao standard (exemplo: CCE-12943-7);
- Fraquezas no software -lista de todas as fraquezas associadas às vulnerabilidades detetadas, através da correspondência entre CVE e CWE.

3.2 Preventive Vulnerability Scanner

Ao mesmo tempo que decorria o desenvolvimento a aplicação, foi decidido pela Sparkint a integração da aplicação na sua plataforma de *Ciberintelligence*. Esta plataforma pretende facultar aos utilizadores informações sobre cibersegurança recolhidas a partir de um conjunto pré-selecionado de fontes abertas.

Uma das ferramentas desenvolvidas, permite que um utilizador possa configurar a sua infraestrutura e ter uma melhor perceção dos riscos a que está exposto ou o nível de exposição de parceiros, clientes ou fornecedores e se poderão, ou não, afetar a instituição. Sempre que forem detetadas alterações na lista de vulnerabilidades conhecidas relacionadas com algum dos componentes, será enviado um email e/ou SMS a alertar da alteração. Desta forma os clientes estão permanentemente informados das vulnerabilidades conhecidas que poderão afetar o seu sistema e assim poderem tomar as medidas necessárias para mitigar o potencial risco.

A outra ferramenta presente nesta plataforma, permite aos utilizadores consultar uma base de dados com conteúdos recolhidos através de fontes abertas, permitindo a pesquisa de notícias, anúncios de vulnerabilidades, CVE's e *exploits*.

Após uma análise cuidada sobre o objetivo do estágio, observou-se que seria interessante integrar a aplicação em desenvolvimento no estágio na plataforma em desenvolvimento pela Sparkint, uma vez que muita das informações utilizadas na plataforma poderiam servir para a aplicação, nomeadamente informações sobre vulnerabilidades.

Desta forma, a ideia inicial sobre o funcionamento da aplicação desenvolvida foi alterada, mas mantendo algumas das bases iniciais, não afetando o trabalho já desenvolvido durante o estágio.

As principais alterações realizadas à proposta inicial são:

- Download da aplicação através da plataforma;
- Deteção apenas de vulnerabilidades;
- Arquitetura cliente/servidor;
- Utilização do SGBD *MySQL*;
- Resultados dos testes apresentados na plataforma;
- Utilização dos ficheiros da linguagem OVAL através do repositório oficial do CIS no *GitHub*[3];
- Necessidade de uma ligação à Internet.

3.2.1 Funcionamento da aplicação

Um dos objetivos da Internet é o de fornecer serviços aos utilizadores. Uma das formas do utilizador ter acesso a um serviço que é fornecido por um servidor remoto é a execução de dois programas: um programa Cliente e um programa Servidor. O cliente executa um programa que envia um pedido ao servidor remoto, e este executa outro programa para fornecer o serviço pedido pelo cliente.

Nesta nova abordagem adotada para a aplicação em desenvolvimento ao longo do estágio, o utilizador acede ao site da Sparkint e realiza a autenticação para aceder à plataforma de Ciberintelligence.

Após a autenticação ser bem-sucedida, o utilizador acede ao painel de controlo, para ter acesso a todas as ferramentas adquiridas, entre elas a ferramenta de análise de vulnerabilidades (**Computer Analysis**).

[3]https://github.com/CISecurity/OVALRepo

Na página da ferramenta, o utilizador realiza o download da ferramenta (cliente) para poder executar a aplicação na máquina local na qual pretende realizar a análise de vulnerabilidades(Ver Figura 3.2).

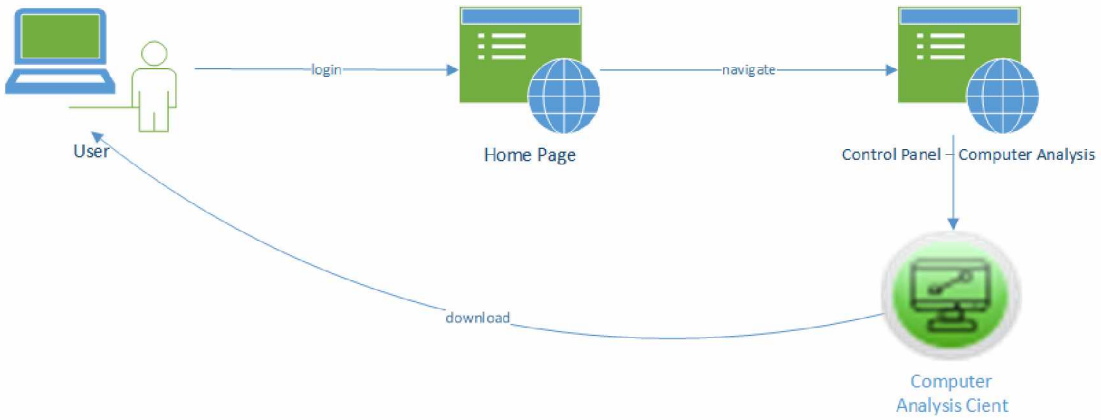


Figura 3.2: Download da aplicação

A Figura 3.3 mostra as informações presentes na página onde é realizado o download da aplicação. Estes requisitos são essenciais para o correto funcionamento da aplicação pois é necessário informar o utilizador de que é necessária uma ligação à Internet e que os resultados da análise de vulnerabilidades serão apresentados no mesmo local onde é realizado o download.

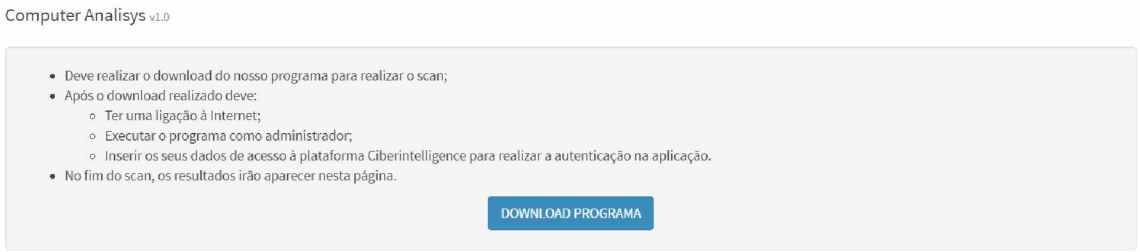


Figura 3.3: Informações importantes sobre a aplicação

Um servidor consiste numa máquina instalada remotamente com o objetivo de prestar serviços aos clientes. Quando este inicia, ele abre o porto para receber os pedidos dos clientes, mas nunca inicia um serviço até ser solicitado para tal.

A figura 3.4 mostra a estrutura da comunicação entre o cliente e o servidor bem como a ligação do servidor com as bases de dados necessárias e também do repositório utilizado.

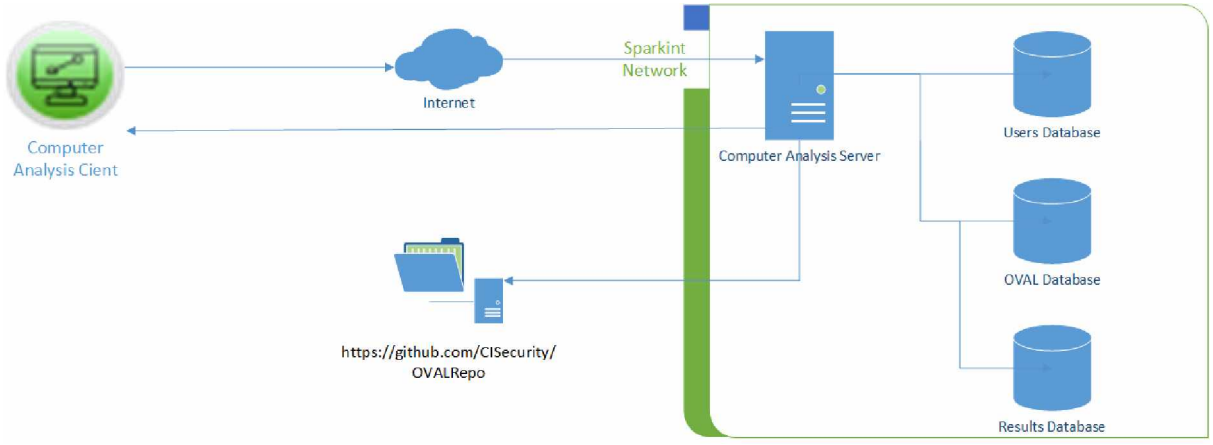


Figura 3.4: Proposta Final da Aplicação

No lado do servidor existem *scripts* desenvolvidos em *Python* para realizar as tarefas de inserção e atualização da base de dados com os ficheiros OVAL, descarregados de forma automática através do repositório oficial da CIS.

Após o *download* da aplicação, o utilizador executa-a (como administrador) e surge uma janela em linha de comandos. O utilizador deve introduzir os seus dados de acesso à plataforma de *Ciberintelligence* da Sparkint.

A aplicação envia os dados para o servidor e este verifica se os dados introduzidos correspondem aos existentes na base de dados dos utilizadores e se for verdade inicia a análise de vulnerabilidades. Caso contrário, informa o utilizador que a autenticação não é válida.

A aplicação inicia então a recolha das aplicações instaladas na máquina onde é executada, enviando de seguida, para o servidor, uma lista com essas aplicações onde consta também a versão do sistema operativo (por exemplo: Windows 10).

O servidor realiza uma correspondência entre o nome das aplicações presentes na máquina local e as aplicações presentes na base de dados com informações da linguagem OVAL. Através desta correspondência, juntamente com a versão do sistema

operativo, o servidor retorna os testes que devem ser realizados no lado do cliente.

O cliente depois de receber as informações do servidor, inicia o processo de realização dos testes para detetar vulnerabilidades nas aplicações instaladas. Após a realização dos testes, o cliente retorna uma lista para o servidor com as informações dos testes cujo resultado foi positivo, ou seja, que na análise de vulnerabilidades, provou-se que a aplicação instalada está realmente vulnerável.

Estes resultados são inseridos na base de dados destinada a armazenar as informações sobre os testes realizados. Importa salientar que apenas as vulnerabilidades classificadas como positivas são inseridas na base de dados.

Na plataforma da Sparkint é apresentada a informação relevante sobre a máquina onde foi realizada a deteção de vulnerabilidades, nomeadamente, o nome da máquina local, a data da realização da deteção, a lista de aplicações detetadas e as vulnerabilidades encontradas durante a deteção. No que diz respeito às vulnerabilidades, é apresentada a identificação da mesma, a classificação de severidade da vulnerabilidade e também, caso existam, *exploits* associados à vulnerabilidade.

Capítulo 4

Implementação da Aplicação

Este capitulo detalha todo o trabalho realizado pelo estagiário ao longo dos seis meses de estágio para a implementação da aplicação. Durante este período foram utilizadas diversas tecnologias e ferramentas (todas elas *open source* ou gratuitas) necessárias para proceder ao desenvolvimento da solução proposta tal como indicado no ponto 3.2 deste relatório. Contudo, algumas das ferramentas utilizadas neste projeto, foram desenvolvidas pela Sparkint, nomeadamente as bases de dados com informações sobre vulnerabilidades, *exploits* e utilizadores.

4.1 Tecnologias e Ferramentas utilizadas

De seguida são apresentadas as tecnologias e ferramentas utilizadas incluindo uma breve descrição das mesmas e também a metodologia utilizada para a realização de algumas tarefas inerentes à solução desenvolvida.

4.1.1 GitHub

O GitHub é um serviço web que oferece diversas funcionalidades extras aplicadas ao git[1]. É uma ferramenta *open source* que permite alojar projetos pessoais ou de equipa. Quase todos os projetos/frameworks/bibliotecas desenvolvidos de forma open source encontram-se no GitHub. É possível acompanhar correções e atualizações realizadas a estes projetos. Esta ferramenta foi utilizada para aceder ao repositório da linguagem OVAL.

[1]Git é um sistema de controlo de versões de ficheiros. Através dele é possível desenvolver projetos em equipa no qual diversas pessoas podem contribuir simultaneamente.

4.1.2 MySQL

O MySQL é um sistema de gestão de bases de dados (SGBD), que utiliza a linguagem SQL (*Structured Query Language*) como interface. É atualmente um dos sistemas mais populares, com mais de 10 milhões de instalações pelo mundo. Este sistema apresenta um excelente desempenho e estabilidade e é uma ferramenta *open source* distribuído pela licença GPL. Devido à grande compatibilidade (suporta conexões com Python, C#, PHP, entre outras) e a pouca exigência de recursos a nível de hardware, foi a escolha óbvia para gerir as bases de dados criadas ao longo deste projeto.

4.1.3 phpMyAdmin

O phpMyAdmin é uma ferramenta de software livre, desenvolvido em PHP, destinada à administração do MySQL na web. Suporta uma vasta gama de operações em MySQL e MariaDB. As operações mais utilizadas (gestão de bases de dados, tabelas, colunas, relações, índices, utilizadores, permissões, etc.) podem ser realizadas através da interface do utilizador e também executar instruções em SQL. A interface gráfica é bastante intuitiva, facilitando a administração de todas as tarefas relacionadas com a base de dados e permitindo a exportação de dados para vários formatos (CSV, SQL, XML, PDF).

4.1.4 PyCharm Edu

O PyCharm é um ambiente de desenvolvimento integrado, em inglês IDE (*Integrated Development Environment*), para a linguagem de programação Python. Foi utilizada a versão Edu uma vez que esta versão não é paga e apresenta os recursos necessários para realizar o desenvolvimento da aplicação. Este IDE apresenta algumas características bastante interessantes como por exemplo, a interação com vários sistemas de controlo de versões (nomeadamente o GitHub), histórico das alterações realizadas a um ficheiro, utilização da guia de estilos para programar em Python (PEP 8[2]). A escolha deste IDE baseou-se apenas por este ser dedicado à linguagem de programação Python.

[2] `https://www.python.org/dev/peps/pep-0008/`

4.1.5 PyInstaller

O PyInstaller é um programa que transforma *scripts* em Python em executáveis *stand-alone*. A suas principais vantagens em relação a ferramentas similares é a possibilidade de trabalhar com ambas as versões do Python (2.7, 3.3-3.5), constrói executáveis de menor tamanho devido à sua compressão transparente, é totalmente multi plataforma e recorre aos sistemas operativos para carregar as bibliotecas. O principal objetivo do PyInstaller é a integração de todos os pacotes necessários para realizar a compressão dos executáveis sem o utilizador precisar de recorrer a "truques"ou ajudas para gerar os executáveis. Uma vez que a aplicação funciona com a arquitetura cliente/servidor, esta ferramenta foi muito útil para desenvolver o cliente.

4.1.6 Python

Python é uma linguagem de altíssimo nível (VHLL - *Very High Level Language*), de sintaxe moderna, orientada a objetos, interpretada via *bytecode*, com tipagem forte (não há conversões automáticas) e dinâmica (não há declaração de variáveis e elas podem conter diferentes objetos), modular, multi plataforma, de fácil aprendizado e de implementação livre. É uma linguagem de uso geral que pode ser empregada em vários tipos de problemas. A biblioteca padrão inclui módulos para processamento de texto e expressões regulares, protocolos de rede (HTTP, FTP, SMTP, POP, XML-RPC, IMAP), acesso aos serviços do sistema operativo, criptografia, interface gráfica, etc. Além da biblioteca padrão, existe uma grande variedade de extensões adicionais para todo tipo de aplicações. Para o desenvolvimento da aplicação foi utilizada a versão 3 desta linguagem.

4.1.7 RPyC

O RPyC ou **R**emote **Py**thon **C**all é uma biblioteca de Python para chamadas remotas de procedimentos (RPC - ver Figura 4.1) e computação distribuída. Ao contrário de outros mecanismo de RPC, esta biblioteca é transparente, simétrica e utiliza uma técnica denominada *object-proxying* que utiliza a natureza dinâmica do Python para ultrapassar as limitações físicas entre processos e computadores, de modo a que os objetos remotos sejam manipulados como se fossem locais. Esta biblioteca foi utilizada para executar as tarefas de trocas de informações entre o cliente e o servidor.

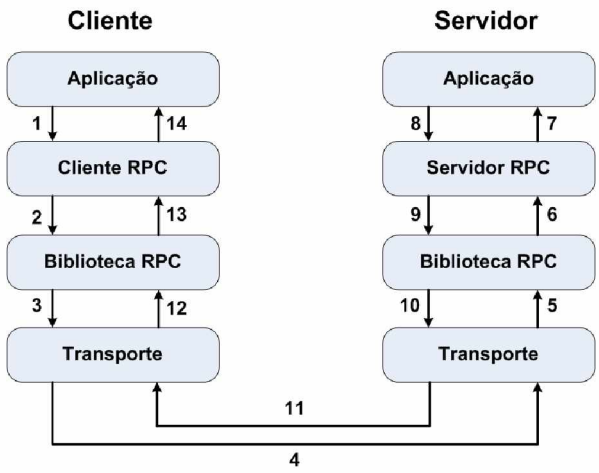


Figura 4.1: Sequência de passos de uma chamada remota de procedimento.

4.2 Estrutura do Repositório OVAL

Antes de abordar a implementação das duas partes que fazem parte da aplicação desenvolvida, é de grande importância referir a estrutura do repositório OVAL bem como as características dos ficheiros que o compõe.

Neste repositório encontram-se todos os ficheiros XML necessários para realizar testes de conformidade a um determinado sistema. Estes testes de conformidade podem ser de cinco classes, Vulnerabilidades, Inventário, Conformidade, Correções e Vários.

O repositório é composto por cinco pastas tal como mostra a Figura 4.2 e cada uma delas, representa um tipo de ficheiro associado a um componente das Definições OVAL.

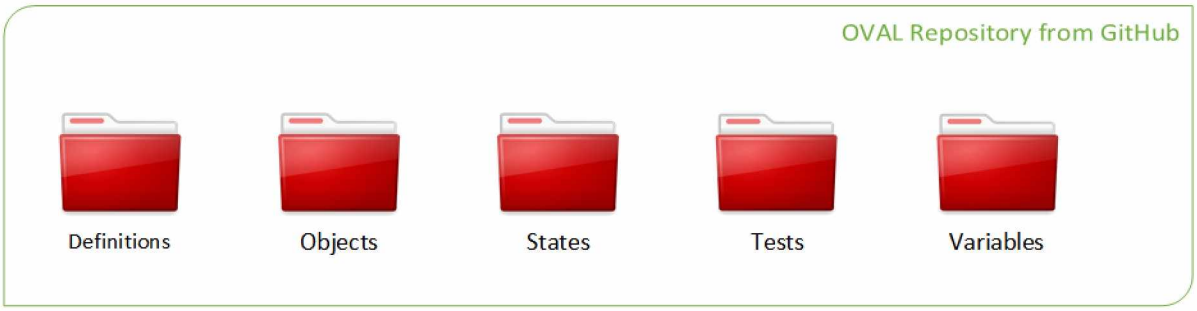


Figura 4.2: Estrutura do Repositório

Uma Definição OVAL é um conjunto de informações padronizadas que permitem realizar testes de conformidade num determinado sistema. Estas definições podem representar uma vulnerabilidade de uma aplicação ou sistema operativo, a presenta de uma aplicação instalada, ou qualquer outro estado especifico de um determinado sistema. Através do conteúdo destas definições, é possível estabelecer processos e procedimentos para realizar testes de conformidade a um sistema.

A Figura 4.3 documenta em maior detalhe os componentes e estrutura de uma Definição OVAL [23]. Os retângulos na figura representam propriedades da definição. As formas circulares representam outros componentes associados à Definição OVAL. As linhas do diagrama representam as relações entre estes componentes. Os componentes adicionais estão assinalados com um asterisco. Além dos componentes presentes na figura acima citada, existe um outro, as Variáveis, que podem ser associadas a estes componentes para representar valores.

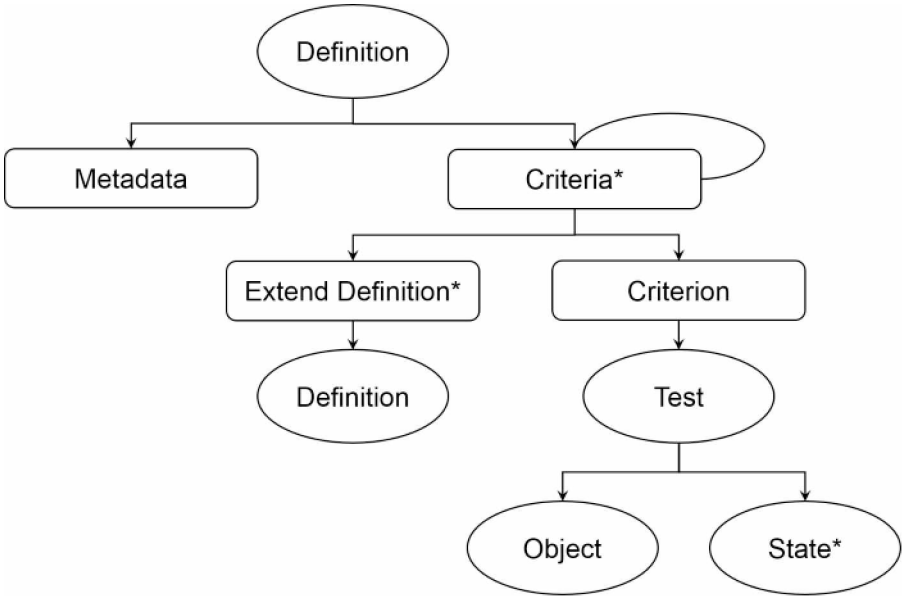


Figura 4.3: Estrutura de uma Definição OVAL

4.3 Processo de Desenvolvimento do Servidor

As operações necessárias para implementar o servidor foram realizadas numa máquina com o sistema operativo *Ubuntu 16.04 LTS 64-bits* com 4GB de memória RAM e um processador *quad-core*. Estas operações foram divididas em duas partes, uma para realizar todas as operações correspondentes à base de dados com as informações provenientes dos ficheiros XML e outra para realizar a ligação com o cliente e as sucessivas trocas de informações entre este e o servidor bem como a inserção dos resultados obtidos na base de dados. Além destas duas operações, também foi implementada uma rotina para atualizar de forma automática o repositório com os ficheiros OVAL. A base de dados foi criada utilizando o sistema de gestão de base de dados MySQL e todas as tabelas foram criadas através do software phpMyAdmin.

Após a atualização do repositório, é criado um ficheiro com informações sobre os vários ficheiros que foram adicionados, removidos ou alterados desde a última atualização. Isto permite que a atualização da base de dados seja realizada afetando apenas os componentes alterados, não sendo necessária a inserção de todos os ficheiros existentes no repositório.

4.3.1 Database

É nesta pasta que se encontram todos os ficheiros (ver Figura 4.4) necessários para transformar os ficheiros em XML numa única base de dados com toda a informação relevante para a troca de informações entre o cliente e o servidor. As duas grandes tarefas relacionadas com a base de dados são: a introdução da informação na base de dados *OVAL Database* e a atualização da mesma com as novas informações provenientes do repositório OVAL.

Os ficheiros para inserção e atualização recorrem aos *scripts* armazenados na pasta *Insert_Files*. Desta forma, fica mais fácil a organização das tarefas realizadas sobre a base de dados. Cada *script* apenas inserir informações nas tabelas da base de dados relacionadas com os componentes que compõe o repositório. A atualização da base de dados é realizada através das informações armazenadas no ficheiro com as modificações ao repositório, removendo e voltando a inserir os componentes presentes no ficheiro.

Esta organização dos ficheiros, facilita a correção e isolamento de erros que possam ocorrer durante a execução do código, permite uma maior facilidade caso seja

necessário realizar modificações ao código e também adicionar novas informações na base de dados relacionadas com os componentes de uma Definição OVAL. Apenas foram utilizados ficheiros relacionados com as definições sobre vulnerabilidades e inventário.

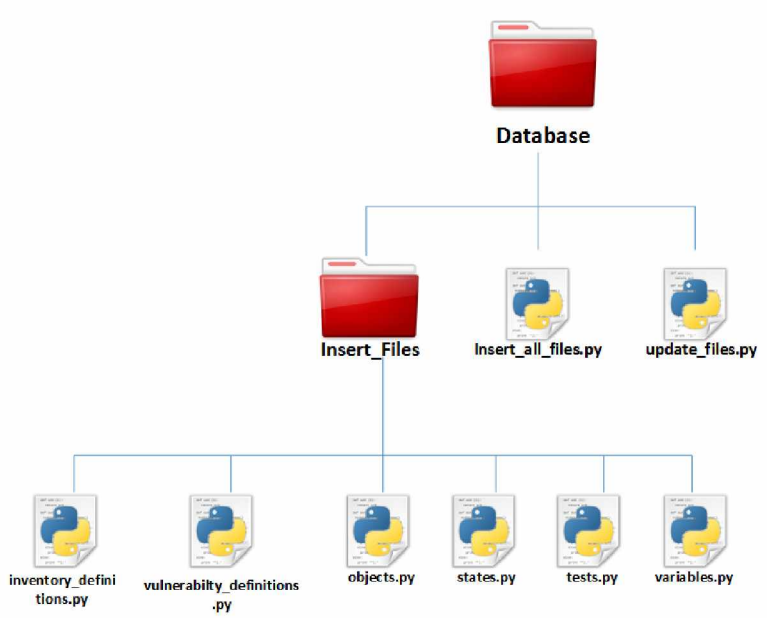


Figura 4.4: Estrutura dos ficheiros na pasta Database

4.3.2 Server

Esta parte da aplicação é composta por cinco ficheiros, agrupados em três tarefas a serem realizadas pelo servidor: estabelecer a ligação e a troca de informações com o cliente, criar um documento XML (como *string* e não como ficheiro propriamente dito) com todos os testes que devem ser realizados na máquina onde o cliente é executado e inserir os resultados dos testes na base de dados. A Figura 4.5 mostra a estrutura dos ficheiros que compõe a pasta com as tarefas realizadas pelo servidor.

O ficheiro *server.py* é o responsável por aceitar as ligações provenientes dos clientes, verificar se o utilizador tem permissões para executar uma análise de vulnerabilidades, ou seja, se este é um utilizador registado na plataforma da Sparkint. Se a autenticação for bem-sucedida, o servidor recebe a lista de aplicações enviadas pelo cliente e realiza um *matching* entre as aplicações presentes nas definições OVAL de inventário e a lista de aplicações do cliente para posteriormente realizar uma pesquisa na base de dados por todas as definições OVAL de vulnerabilidades

associadas às aplicações e ao sistema operativo da máquina onde é executado o cliente da aplicação.

No ficheiro *create_xml.py* é criado uma string como se fosse um ficheiro XML com as informações dos testes a serem realizados pelo cliente para analisar a presença de vulnerabilidades na lista de aplicações instaladas na máquina onde o cliente é executado. Através do módulo *lxml*[3] e da ligação à base de dados é então criado o XML e enviado para o cliente. Devido à existência de vários tipos de objetos nas definições OVAL e dado que nesta aplicação apenas são realizados testes a ficheiros e ao registo do Windows, optou-se por separar a introdução destes elementos no XML através de dois ficheiros distintos, *file_objects.py* e *registry_objects.py*, pois estes dois tipos de objetos apresentam características diferentes e podem incluir informações de ambos os tipos de objetos.

Por último, o ficheiro *insert_results.py* é o responsável por inserir os resultados dos testes, na base de dados *Results*, para estes serem apresentados na plataforma da Sparkint. Tal como na criação dos testes, esta tarefa recebe um documento XML no formato de string proveniente do cliente e realiza o *parsing* do mesmo para obter as informações necessárias.

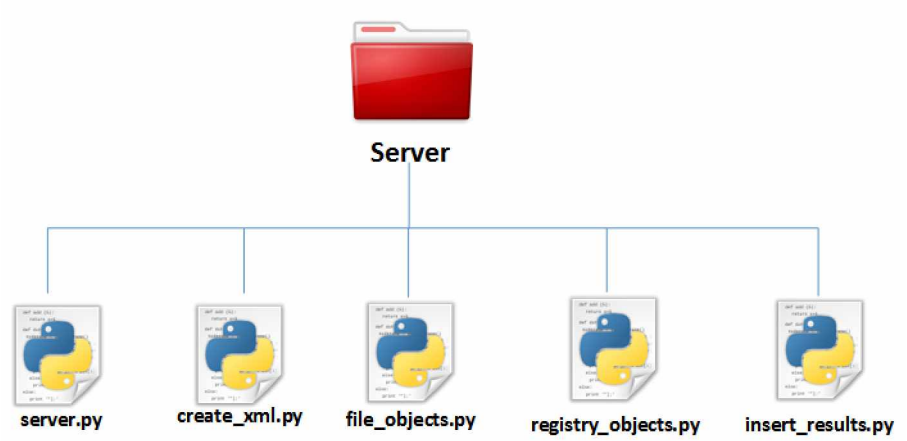


Figura 4.5: Estrutura dos ficheiros na pasta Server

[3] `http://lxml.de/`

4.4 Processo de Desenvolvimento do Cliente

O desenvolvimento do cliente desta aplicação, foi realizado em duas máquinas, uma com o sistema operativo *Windows 10 64-bits* e outra com o *Windows 7 32-bits*. Tal como acontece no servidor, todos os ficheiros necessários foram desenvolvidos através da linguagem de programação Python e de algumas bibliotecas desta linguagem.A Figura 4.6 mostra a estrutura dos ficheiros que compõe o Cliente desta aplicação.

Esta parte da aplicação é responsável por encontrar as aplicações instaladas na máquina onde é executado o cliente, realizar os testes de conformidade para detetar a presença de vulnerabilidades conhecidas nas aplicações instaladas e realizar a troca de informações com o servidor.

O ficheiro *installed_programs.py* realiza a deteção das aplicações na máquina onde é executado o cliente e cria uma lista com as mesmas. Nesta versão inicial da aplicação, a deteção é realizada apenas através do registo do Windows, localizando todas as aplicações instaladas no caminho do registo onde as mesmas são instaladas dependendo a arquitetura do sistema operativo[4]. Nesta procura por aplicações, são excluídas as atualizações do sistema operativo bem como as ferramentas (*tools*) associadas a uma aplicação.

O ficheiro *computer_analysis.py* representa o ficheiro principal de toda a aplicação onde é realizada a ligação com o servidor e o envio das informações para o mesmo, a criação do documento XML com os resultados dos testes que serão enviados para o servidor e "*interface gráfica*"de interação com o utilizador.

O cliente inicia o processo de ligação com o servidor através do módulo **RPyC**, enviando como parâmetros o IP e porto do servidor ao qual se pretende ligar, e utiliza o método criado no servidor para realizar a autenticação do utilizador (tal como indicado anteriormente, o utilizador deve fornecer os seus dados de acesso à plataforma da Sparkint). Após a correta autenticação do utilizador, o servidor retorna a identificação do utilizador para ser posteriormente utilizada. É enviado para o servidor o nome da máquina local, a data atual, o sistema operativo e a lista de aplicações instaladas na máquina. Se o utilizador não for autenticado, é mostrada a informação

[4]Nos sistemas operativos de 64-bits encontram-se em *SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall* e nos de 32-bits em *SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall*

de autenticação inválida e o utilizador pode voltar, ou não, a realizar a autenticação.

O servidor retorna as informações de todos os testes relacionados com as vulnerabilidades associadas às aplicações e ao sistema operativo e é utilizado o ficheiro *execute_tests.py* para dar inicio à deteção de vulnerabilidades. A primeira tarefa a ser realizada é o *parsing* da *string* com o documento em XML cujo conteúdo representa as condições e testes a serem realizados para verificar o estado de conformidade das aplicações instaladas. Tal como referido anteriormente, apenas são realizados testes a nível de ficheiros e do registo do Windows. Estes testes, estão devidamente identificados no documento XML enviado pelo servidor. Se o teste for do tipo de ficheiros (*file tests*) é realizado o teste através dos métodos presentes no ficheiro *execute_tests.py*, caso contrário é necessária a utilização do ficheiro *registry_tests.py* que é o responsável por realizar os testes de vulnerabilidades a nível do registo do Windows (*registry tests*).

Por último, os resultados dos testes cujo resultado é verdade, ou seja, nos quais ficou provado que a aplicação é vulnerável, são devolvidos para um método presente no ficheiro *computer_analysis.py*, que cria um documento XML com esses mesmo resultados e com as informações recolhidas inicialmente (nome do computador, data e identificação do utilizador). Este documento é enviado para o servidor e os resultados são inseridos na base de dados para serem apresentados na plataforma da Sparkint.

O executável do cliente foi criado no máquina com o sistema operativo *Windows 7 32-bits* através da ferramenta **PyInstaller**, que agrupa todos os ficheiros da estrutura do cliente num único ficheiro de extensão *.exe*, permitindo assim realizar o download de apenas um ficheiro na plataforma da Sparkint. Este ficheiro deve ser executado como a administrador para obter os melhores resultados na deteção de vulnerabilidades.

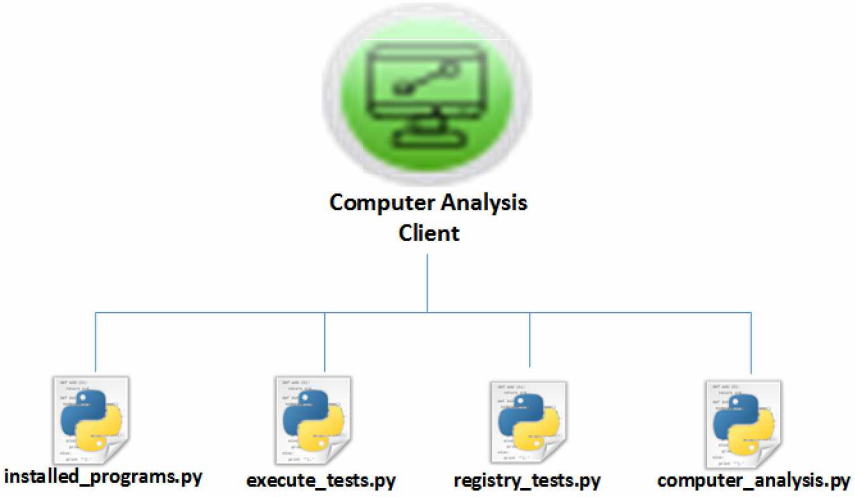


Figura 4.6: Estrutura dos ficheiros que compõem o Cliente

Capítulo 5

Avaliação

A avaliação desta aplicação passa necessariamente por testar o seu funcionamento e a sua eficácia tendo conta o seu objetivo principal, a deteção de vulnerabilidades em aplicações instaladas num sistema operativo da família Windows. Os testes realizados nesta fase, além de cumprirem com o objetivo proposto ao longo do estágio, também fornecem informações sobre algumas falhas detetadas, possíveis melhorias e alterações a serem implementadas na aplicação num trabalho futuro.

Para a realização dos testes de avaliação do funcionamento da aplicação e com o objetivo de abranger diferentes versões dos sistemas operativos Windows, foram utilizadas duas máquinas virtuais, uma com o sistema operativo *Windows 7 32-bits* (WIN7Vul) e outra com o *Windows 8.1 64-bits* (WIN8Vul). Em cada uma destas máquinas foram instaladas várias aplicações, das quais algumas apresentam vulnerabilidades conhecidas, de acordo com a informação presente no site oficial da linguagem OVAL e outras sem qualquer tipo de vulnerabilidades.

Nos testes realizados optou-se por excluir a instalação de *browsers* nas máquinas, pois quando estes são executados pela primeira vez, caso a versão instaladas seja anterior à versão atual, o próprio programa realiza a atualização do software dificultado a deteção de vulnerabilidades ou induzindo em erro a mesma deteção. Apesar de este tipo de software ser aquele que apresenta um maior número de vulnerabilidades conhecidas, achamos que seria melhor apresentar menos resultados mas com uma maior credibilidade dos mesmos.

5.1 Metodologia utilizada

Antes da realização dos testes efetuou-se uma preparação dos mesmos, ou seja, foi necessário encontrar aplicações com versões vulneráveis, sabendo assim à partida que a aplicação desenvolvida teria que apresentar vulnerabilidades quando executada.

A escolha das versões das aplicações a testar foi realizada após consulta do site oficial da linguagem OVAL, onde é possível obter definições de vulnerabilidades de várias aplicações existentes.

A Figura 5.1 exemplifica uma das consultas realizadas no repositório OVAL, em que foram consultadas as definições de vulnerabilidades da aplicação *VLC Media Player*, quando esta se encontra instalada no sistema operativo *Windows 8.1*. A pesquisa em questão retornou três definições de vulnerabilidades (Figura 5.2), foram analisados esses três resultados, sendo assim possível verificar quais as versões da aplicação *VLC Media Player* vulneráveis. Tendo já essa informação fez-se o download e a consequente instalação do software em questão, para uma máquina virtual de forma a poder executar a aplicação desenvolvida ao longo do estágio e observar os resultados obtidos na plataforma da Sparkint.

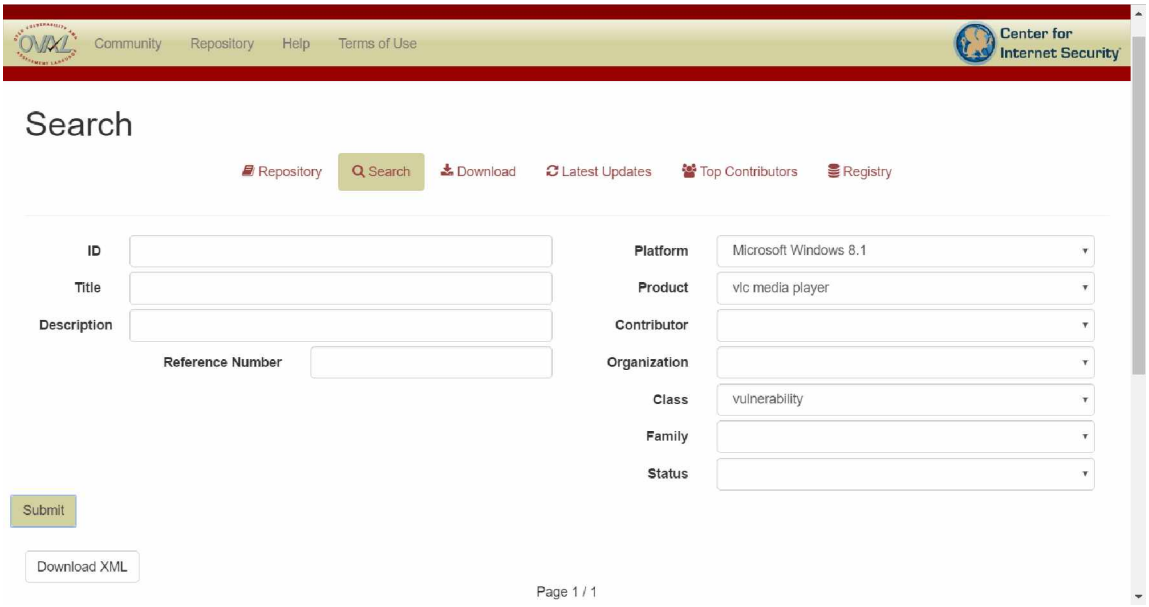
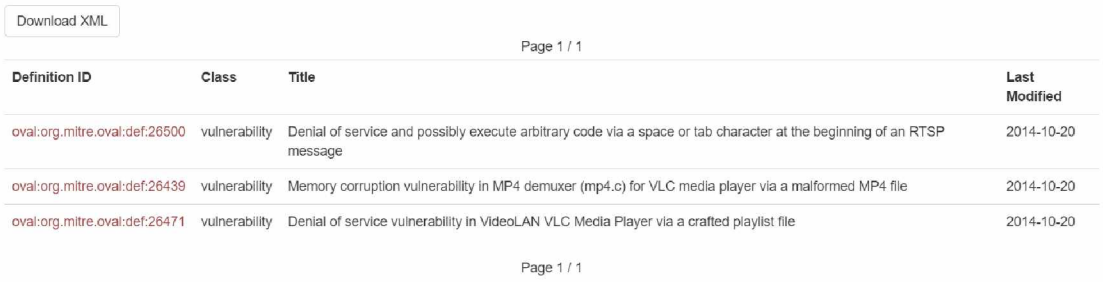


Figura 5.1: Pesquisa de Vulnerabilidades



Download XML

Page 1 / 1

Definition ID	Class	Title	Last Modified
oval:org.mitre.oval:def:26500	vulnerability	Denial of service and possibly execute arbitrary code via a space or tab character at the beginning of an RTSP message	2014-10-20
oval:org.mitre.oval:def:26439	vulnerability	Memory corruption vulnerability in MP4 demuxer (mp4.c) for VLC media player via a malformed MP4 file	2014-10-20
oval:org.mitre.oval:def:26471	vulnerability	Denial of service vulnerability in VideoLAN VLC Media Player via a crafted playlist file	2014-10-20

Page 1 / 1

Figura 5.2: Lista de Vulnerabilidades da aplicação *VLC media player*

Realizou-se a deteção e confirmação dessas mesmas vulnerabilidades de forma manual, ou seja, as informações sobre os testes (presentes nos ficheiros da linguagem OVAL) que a aplicação deveria realizar para confirmar a presença de uma vulnerabilidade, e os resultados dos mesmos, foram confirmados verificando todos os passos realizados pela aplicação. Criou-se um ficheiro de apoio para identificar os testes realizados e os resultados obtidos, facilitando assim, a observação do funcionamento da aplicação.

5.2 Avaliação do Funcionamento

Para avaliar o funcionamento da aplicação desenvolvida foram utilizadas duas máquinas virtuais com dois sistemas operativos diferentes, de forma a avaliar o desempenho em sistemas com arquiteturas diferentes. Os testes realizados consistiram na instalação de algumas aplicações, em que uma delas apresentava uma série de vulnerabilidades de acordo com a pesquisa realizada no site oficial da linguagem OVAL. O procedimento para realizar os testes em ambas as máquinas foi similar, consistiu em instalar as aplicações previamente selecionadas, executar a aplicação desenvolvida e observar os resultados obtidos na plataforma da Sparkint.

5.2.1 Autenticação na plataforma

Antes de realizar os testes de funcionamento, realizou-se um teste para verificar o bom funcionamento da autenticação na plataforma da Sparkint, pois sem esta verificação, os testes nunca iriam ser executados corretamente.

A Figura 5.3 mostra o ecrã inicial quando se executa a aplicação. O utilizador deve fornecer os seus dados de acesso à plataforma da Sparkint para realizar a autenticação.

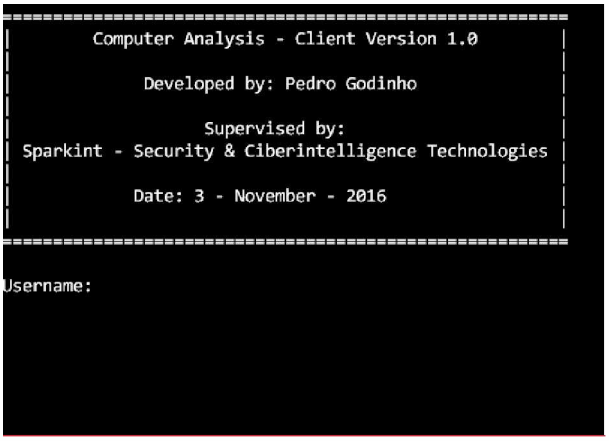


Figura 5.3: Ecrã inicial da aplicação

Se a autenticação for realizada com sucesso, a aplicação inicia a deteção de vulnerabilidades e informa o utilizador que esta pode demorar alguns minutos (Figura 5.4). Caso contrário, a aplicação informa o utilizador que a autenticação é inválida e pede para este pressionar a tecla Enter (Figura 5.5), voltando ao ecrã principal da aplicação.

Figura 5.4: Autenticação bem sucedida.

Figura 5.5: Autenticação inválida.

Se a aplicação não tiver uma ligação à Internet, o utilizador faz a autenticação e a aplicação fecha a janela automaticamente. Nesta versão da aplicação, não foi adicionada nenhuma forma de tratamento de exceções para lidar com este tipo de situações.

5.2.2 Windows 8.1 64-bits

Esta máquina virtual de testes apresenta uma configuração de *hardware* com 2GB de memória RAM, um processador *quad-core* e um disco rígido de 60GB. Foram instaladas as seguintes aplicações:

- Foxit Reader (versão 8.0.6.909)
- MySQL Server 5.5 (versão 5.5.30)
- Steam (versão 2.10.91.91)
- VLC media player (versão 2.2.4)

Após a análise das definições OVAL associadas a estas aplicações, constatou-se que das aplicações instaladas a melhor candidata a realizar o teste de análise de vulnerabilidades seria a aplicação **MySQL Server 5.5**.

Esta aplicação apresenta uma lista de dez vulnerabilidades conhecidas, associadas a esta versão especifica. A Figura 5.6 mostra a lista de vulnerabilidades e definições OVAL obtidas através de uma pesquisa por sistema operativo e aplicação no site oficial da linguagem OVAL.

Definition ID	Class	Title	Last Modified
oval:org.cisecurity:def:729	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.47 and earlier 5.6.28 and earlier and 5.7.10 and earlier (CVE-2016-0644)	2016-07-01
oval:org.cisecurity:def:715	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.47 and earlier 5.6.28 and earlier and 5.7.10 and earlier (CVE-2016-0640)	2016-07-01
oval:org.cisecurity:def:730	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.48 and earlier 5.6.29 and earlier and 5.7.11 and earlier (CVE-2016-0647)	2016-07-01
oval:org.cisecurity:def:724	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.47 and earlier 5.6.28 and earlier and 5.7.10 and earlier (CVE-2016-0646)	2016-07-01
oval:org.cisecurity:def:1315	vulnerability	Unspecified vulnerability in Oracle MySQL 5.6.30 and earlier and 5.7.12 and earlier – CVE-2016-3614	2016-11-25
oval:org.cisecurity:def:1305	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.49 and earlier 5.6.30 and earlier and 5.7.12 and earlier and MariaDB before 5.5.50 10.0.x before 10.0.26 and 10.1.x before 10.1.15 - CVE-2016-5440	2016-11-25
oval:org.cisecurity:def:1302	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.48 and earlier 5.6.29 and earlier and 5.7.11 and earlier and MariaDB before 5.5.49 10.0.x before 10.0.25 and 10.1.x before 10.1.14 - CVE-2016-5444	2016-11-25
oval:org.cisecurity:def:1316	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.49 and earlier 5.6.30 and earlier and 5.7.12 and earlier and MariaDB before 5.5.50 10.0.x before 10.0.26 and 10.1.x before 10.1.15 – CVE-2016-3521	2016-11-25
oval:org.cisecurity:def:1314	vulnerability	Unspecified vulnerability in Oracle MySQL 5.5.49 and earlier 5.6.30 and earlier and 5.7.12 and earlier and MariaDB before 5.5.50 10.0.x before 10.0.26 and 10.1.x before 10.1.15 – CVE-2016-3615	2016-11-25
oval:org.cisecurity:def:1199	vulnerability	Vulnerability in Oracle MySQL through 5.5.52 5.6.x through 5.6.33 and 5.7.x through 5.7.15; MariaDB before 5.5.51 10.0.x before 10.0.27 and 10.1.x before 10.1.17 - CVE-2016-6662	2016-10-28

Figura 5.6: Lista de Vulnerabilidades associadas à aplicação *MySQL Server 5.5*

Apenas as aplicações *Foxit Reader* e *Steam* não apresentam vulnerabilidades associadas quando se encontram instaladas neste sistema operativo. Para a aplicação *VLC media player*, existem três vulnerabilidades associadas, mas nenhuma delas afeta à versão da aplicação instalada nesta máquina de testes.

O passo seguinte foi executar a aplicação desenvolvida para realizar a deteção de vulnerabilidades, tendo em conta que já se tinha conhecimento que a aplicação apenas iria detetar vulnerabilidades na aplicação *MySQL Server 5.5*.

Após a conclusão da deteção de vulnerabilidades, acedeu-se à plataforma da Sparkint para observar os resultados obtidos.

A Figura 5.7 mostra a lista (incompleta) dos resultados obtidos após a análise de vulnerabilidades realizado pela aplicação. No lado esquerdo da imagem, existe um menu *drop-down* com o nome da máquina na qual se realizou a deteção de vulnerabilidades e a data de inicio da mesma. Quando se clica em cima deste menu, surge a lista de aplicações instaladas na máquina de testes. Nesta lista, surge uma aplicação que não se encontra instalada na máquina, pois quando é realizado o *matching* entre a lista de aplicações instaladas e as presentes na base de dados com as informações OVAL, a palavra *MySQL* está contida dentro da aplicação *MySQL Server 5.5* e esta é incluída na lista de aplicações que devem ser testadas.

Os resultados obtidos mostram que das dez vulnerabilidades associadas à aplicação *MySQL Server 5.5*, todas elas foram detetadas pela aplicação desenvolvida. Estes resultados foram analisados manualmente e a aplicação realizou todos os testes relacionados com as definições OVAL associadas a estas vulnerabilidades de forma satisfatória, não apresentando qualquer tipo de falso positivo. Também foram analisados os testes realizados para as definições da aplicação *VLC media player*, e a aplicação mostrou que todas as condições necessárias, para a aplicação ser considerada vulnerável apresentaram resultados negativos.

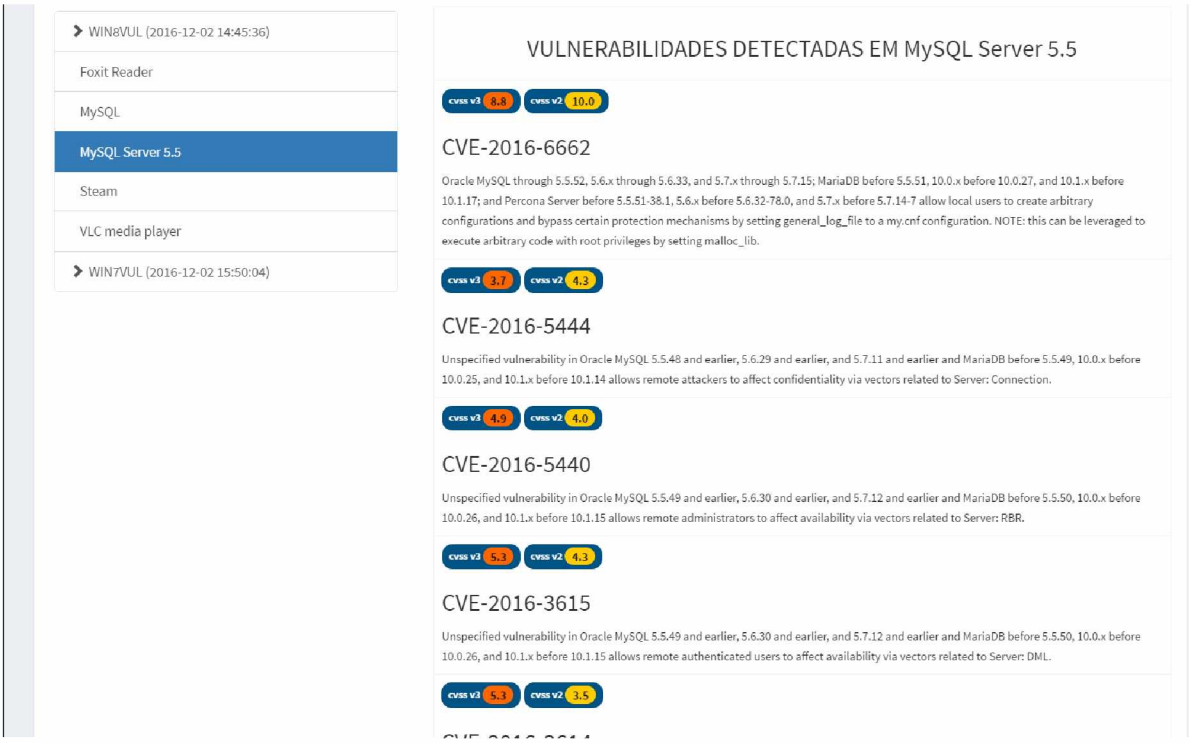


Figura 5.7: Resultados obtidos na máquina WIN8Vul

5.2.3 Windows 7 32-bits

Esta máquina virtual de testes apresenta uma configuração de *hardware* de 2GB de memória RAM, um processador *quad-core* e um disco rígido de 60GB. Optou-se por criar uma máquina virtual com as mesmas características da máquina com o sistema operativo *Windows 8.1* para não existirem diferenças a nível de *hardware*. Foram instaladas as seguintes aplicações:

- Microsoft .NET Framework 4.5
- VLC media player (versão 2.0.1)

Foram analisadas algumas definições OVAL associadas a estas duas aplicações e optou-se por selecionar a aplicação **VLC media player** para realizar o teste de análise de vulnerabilidades. Esta versão da aplicação é já bastante antiga e conjugada com um sistema operativo também já com alguns anos, a probabilidade de apresentar vulnerabilidades é bastante elevada. Escolheu-se esta aplicação por se tratar de um reprodutor de conteúdos multimédia e por este tipo de aplicação ser muito explorada a nível de ataques através de *exploits*.

Esta aplicação apresenta, sessenta definições (ver Figura 5.8) sobre vulnerabilidades, após a pesquisa realizada no repositório oficial da linguagem OVAL.

Page 1 / 2 > >>

Definition ID	Class	Title	Last Modified
oval:org.mitre.oval:def:12190	vulnerability	Untrusted search path vulnerability in bin/winvlc.c in VLC Media Player 1.1.3 and earlier allows DLL hijacking attacks	2012-11-19
oval:org.mitre.oval:def:12415	vulnerability	Denial of service vulnerability in the MKV demuxer plugin in VideoLAN VLC media player in VideoLAN VLC Media Player before 1.1.7	2012-11-19
oval:org.mitre.oval:def:12460	vulnerability	Multiple heap-based buffer overflow vulnerabilities in VideoLAN VLC Media Player before 1.1.6	2012-11-19
oval:org.mitre.oval:def:12414	vulnerability	Heap based memory corruption vulnerability in "StripTags()" function within the USF and Text subtitles decoders in VideoLAN VLC Media Player 1.1 before 1.1.6	2012-11-19
oval:org.mitre.oval:def:14817	vulnerability	Multiple heap-based buffer overflows in VideoLAN VLC media player before 2.0.1	2012-11-19
oval:org.mitre.oval:def:14851	vulnerability	Heap-based buffer overflow in the DemuxAudioSipr function in real.c in the RealMedia demuxer in VideoLAN VLC media player 1.1.x before 1.1.11	2012-11-19
oval:org.mitre.oval:def:14858	vulnerability	Heap-based buffer overflow in the AVI_ChunkRead_strf function in libavi.c in the AVI demuxer in VideoLAN VLC media player before 1.1.11	2012-11-19
oval:org.mitre.oval:def:14774	vulnerability	Integer overflow in the XSPF playlist parser in VideoLAN VLC media player 0.8.5 through 1.1.9	2012-11-19
oval:org.mitre.oval:def:14876	vulnerability	The RTSP module in VideoLAN VLC 0.8.6d allows remote attackers to cause a denial of service	2012-11-19
oval:org.mitre.oval:def:14872	vulnerability	Stack-based buffer overflow in the ParseSSA function (modules/demux/subtitle.c) in VLC 0.8.6e	2012-11-19
oval:org.mitre.oval:def:15540	vulnerability	Vulnerability in libpng, plugin in VideoLAN VLC media player 2.0.3	2012-12-

Figura 5.8: Lista de Vulnerabilidades associadas à aplicação *VLC media player*

Já a aplicação *Microsoft .NET Framework 4.5* quando instalada numa máquina com o sistema operativo *Windows 7*, apresenta trinta vulnerabilidades conhecidas e documentadas através de definições OVAL. Apesar de ser um número considerável de vulnerabilidades, optou-se por utilizar a outra aplicação, tal como foi justificado anteriormente. Como seria de prever, o número de vulnerabilidades é muito superior quando uma aplicação é instalada num sistema operativo mais antigo.

O procedimento realizado nesta máquina de testes foi exatamente igual ao realizado na máquina de testes com o sistema operativo Windows 8.1. Assim sendo, executou-se a aplicação desenvolvida, aguardou-se pelo término da execução e verificou-se os resultados obtidos no site da plataforma.

A Figura 5.9 apresenta os resultados obtidos, na plataforma da Sparkint, após a execução da aplicação desenvolvida.

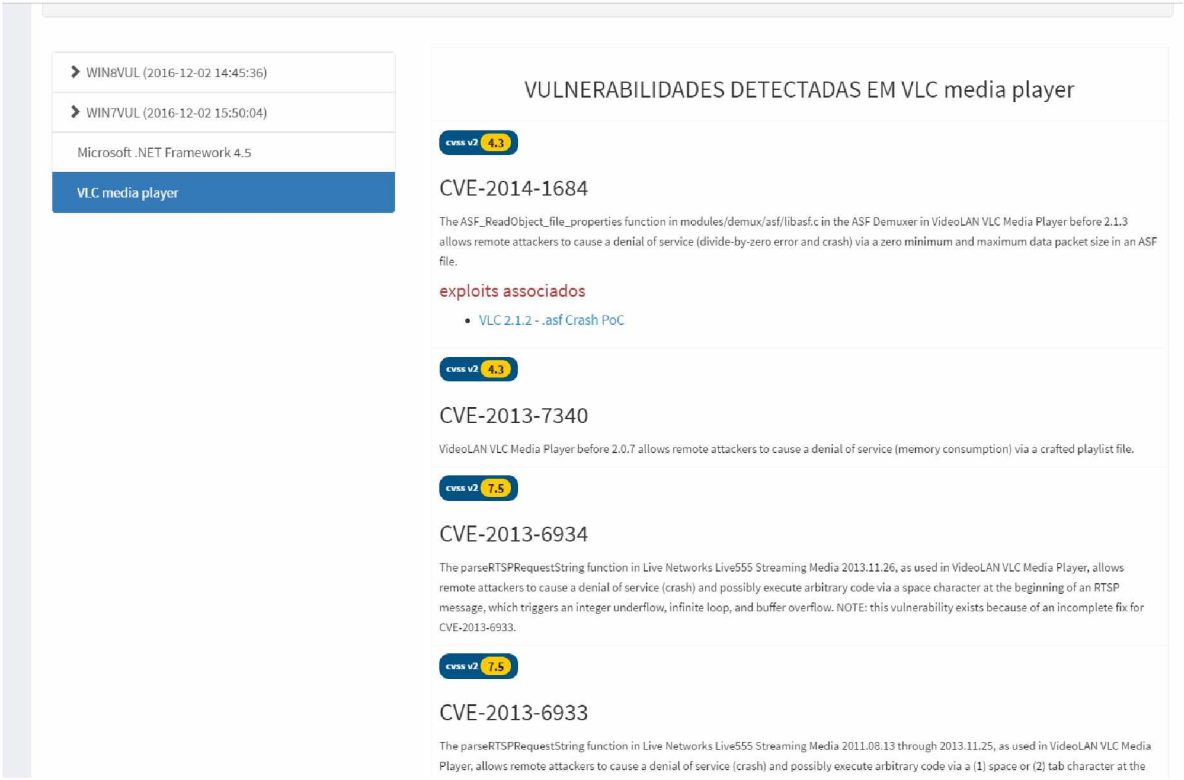


Figura 5.9: Resultados obtidos na máquina WIN7Vul

É possível verificar, pelos resultados, que das sessenta definições associadas à aplicação *VLC media player*, a versão escolhida para realizar este teste encontra-se vulnerável em seis dessas definições. Tal como no teste anterior, os resultados foram analisados manualmente para verificar que estes resultados são verdadeiros e não falsos positivos.

A Figura 5.9 também mostra a existência de *exploits* associados a uma determinada vulnerabilidade (CVE-2014-1684). Este tipo de informação é bastante útil para o utilizador pois apresenta provas de que existem formas de explorar a vulnerabilidade encontrada. Além das informações sobre *exploits*, a plataforma da Sparkint mostra ao utilizador a classificação de severidade de cada uma das vulnerabilidades detetadas. No teste realizado anteriormente, as vulnerabilidades detetadas apresentam as duas versões do sistema de classificação (versão 2 e versão 3).

5.3 Avaliação da aplicação

Apesar da aplicação, desenvolvida durante o período de estágio atingir os objetivos propostos, é de grande importância abordar alguns dos problemas e falhas encontradas ao longo de todas as tarefas realizadas no desenvolvimento da aplicação.

Um dos grandes problemas encontrados no decorrer do estágio foi o reduzido número de vulnerabilidades presentes na linguagem OVAL. Considerando que existem atualmente 79456 vulnerabilidades conhecidas, segundo as informações presentes no site **CVE Details**[1], para todas as aplicações e sistemas operativos existentes no mercado (ver Figura 5.10). A linguagem OVAL, neste momento, contém 13812 definições sobre vulnerabilidades, representando 17.4% do total de todas as vulnerabilidades conhecidas atualmente.

Distribution of all vulnerabilities by CVSS Scores

CVSS Score	Number Of Vulnerabilities	Percentage
0-1	80	0.10
1-2	612	0.80
2-3	3238	4.10
3-4	2000	2.50
4-5	15714	19.80
5-6	15741	19.80
6-7	9797	12.30
7-8	19937	25.10
8-9	347	0.40
9-10	11990	15.10
Total	79456	

Weighted Average CVSS Score: **6.8**

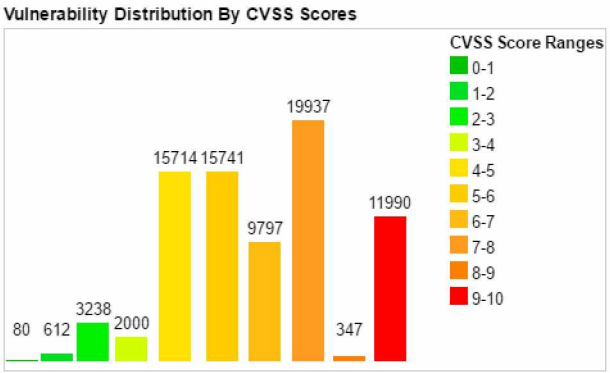


Figura 5.10: Distribuição das classificações de Vulnerabilidades

Estes dados indicam que a aplicação desenvolvida não consegue realizar testes a todas as vulnerabilidades conhecidas. Este problema limita, de certa forma, as informações que são transmitidas ao utilizador final, não apresentando resultados tão completos como o utilizador poderia esperar antes de executar a aplicação. Não podemos provar ou afirmar, com total certeza, que algumas das aplicações instaladas numa determinada máquina, cujos resultados não aparecem na plataforma da Sparkint, não apresentem vulnerabilidades associadas, pois na base de dados desenvolvida com as informações para realizar os testes necessários, não se encontram as informações sobre vulnerabilidades ligadas à aplicação em causa.

[1] https://www.cvedetails.com

Os ficheiros que contêm definições OVAL, para uma determinada vulnerabilidade, apresentam uma lista de produtos (aplicações) e plataformas (sistemas operativos) nos quais essa vulnerabilidade pode ser detetada. Se estes tipos de ficheiros não forem atualizados de forma periódica, ou quando uma nova plataforma surge no mercado, corre-se o risco de que uma vulnerabilidade não seja testada nessa plataforma. Vamos dar como exemplo umas das aplicações testadas neste projeto, o leitor multimédia *VLC media player*. Segundo as informações presentes nos ficheiros das definições OVAL, esta aplicação apenas é vulnerável nos sistemas operativos anteriores ao Windows 10. Contudo, os testes realizados nos outros sistemas operativos em nada são diferentes daqueles que seriam realizados no Windows 10.

Será que a aplicação não é vulnerável neste sistema operativo?

As informações presentes na *National Vulnerabilty Database* dizem que a aplicação pode ser eventualmente vulnerável, pois não há nenhuma referência aos sistemas operativos nos quais a aplicação é vulnerável, mas sim à aplicação em si (Figura 5.11). Este exemplo não se aplica apenas a esta aplicação, mas a muitas outras, devido ao facto de o número de vulnerabilidades presentes na linguagem OVAL ser bastante menor do que o número de vulnerabilidades conhecidas atualmente, e como os ficheiros são mantidos por uma comunidade, nem sempre é possível atualizar os ficheiros da linguagem com a rapidez necessária para evitar este tipo de situações.

Figura 5.11: Versões vulneráveis da aplicação *VLC media player*

Neste momento, na aplicação implementada existe uma pequena falha que durante o desenvolvimento da aplicação não foi contemplada podendo apresentar alguns problemas quando é realizada a deteção de vulnerabilidades. Tendo em conta que a aplicação foi projetada recorrendo à arquitetura cliente/servidor, se a ligação entre o cliente e o servidor por interrompida por algum motivo (falha na ligação à Internet, término abrupto da execução da aplicação ou outro fator que influencie a ligação entre ambos), os resultados não serão enviados para a plataforma da Sparkint, sendo necessário realizar outra execução da aplicação para que esta termine sem problemas sendo os resultados apresentados na plataforma.

O tempo de execução da aplicação é proporcional ao número de aplicações instaladas e ao número de definições de vulnerabilidades associadas às mesmas. Quanto maior for o número de definições, maior será o número de testes de conformidade que a aplicação terá que realizar, aumentando assim o tempo de execução. A ligação à Internet também pode influenciar o desempenho da aplicação pois uma ligação com uma taxa de transferência de dados menor, aumenta o tempo da troca de informações entre o cliente e o servidor. Neste caso o impacto no desempenho da aplicação não será tão grande como no caso anterior, se o tamanho da informação a ser trocada não for muito elevado.

Nesta fase, a aplicação deve ser encarada como um protótipo e não como um produto final já terminado, uma vez que existem algumas melhorias que devem ser acrescentadas à aplicação para melhorar a mesma, quer a nível de desempenho quer a nível de resultados obtidos nos testes.

A deteção do software instalado numa determinada máquina, apenas é verificado numa localização especifica do registo do Windows, tal como referido no ponto 4.3. Esta deteção é bastante limitada, uma vez que podem existir aplicações instaladas noutras localizações, tanto a nível do registo como a nível de ficheiros no sistema operativo, que não são devidamente detetadas pela aplicação. Esta limitação da aplicação, limita o número de definições OVAL a serem utilizadas, os testes para detetar vulnerabilidades, e também os resultados obtidos.

Devido à existência de muitas variáveis no universo dos ficheiros em XML da linguagem OVAL, é possível que existam algumas pequenas falhas na realização dos testes de conformidade por parte da aplicação desenvolvida. Por exemplo, existem cerca de 87.000 *Criterias* e 125.000 *Criterions* diferentes que podem ser conjugados para formar uma definição OVAL, juntamente com os outros elementos (objetos, testes, estados e variáveis). Trabalhar com um número tão elevado de variáveis e componentes desta linguagem, associado ao pouco tempo disponível para desenvolver a aplicação, pode levar ao aparecimento de erros e falhas não contabilizados na fase de análise e implementação da solução proposta.

Capítulo 6

Conclusão e Trabalho Futuro

No estágio realizando na empresa Sparkint – Security & Ciberintelligence Technologies, Lda., desenvolveu-se uma aplicação cujos principais objetivos são a deteção de vulnerabilidades em aplicações instaladas em sistemas operativos da família Windows e a apresentação de informações relevantes sobre as mesmas numa plataforma.

O desenvolvimento desta aplicação realizou-se recorrendo exclusivamente a ferramentas gratuitas e de utilização livre. Desde o sistema operativo do servidor, passando pela linguagem de programação e bibliotecas utilizadas até à linguagem para realizar os testes de conformidade de um determinado sistema. Esta aplicação apresenta uma arquitetura cliente/servidor. O utilizador realiza o download do cliente na plataforma da Sparkint, executa-o como administrador, efetua a autenticação e após a deteção de vulnerabilidades terminar, acede à plataforma para observar os resultados obtidos.

O cliente interage com o servidor através de chamadas remotas de procedimentos (RPC) para poder utilizar objetos remotos como se estes fossem tratados localmente, realiza uma constante troca de informações para aceder à base de dados presente no servidor e procede à realização dos testes de conformidade, inserido os resultados noutra base de dados.

O servidor aloja as bases de dados necessárias para a troca de informações com o cliente e utiliza um clone do repositório oficial da linguagem OVAL com ficheiros em XML. Também é responsável por atualizar o repositório e inserir as alterações na base de dados, mantendo assim as informações sobre vulnerabilidades o mais atualizado possível. A principal função do servidor é estabelecer as ligações com

os clientes e fornecer as informações necessárias para cada um, permitindo assim a realização dos testes de conformidade, nas máquinas onde os clientes são executados.

De forma a atingir os objetivos propostos, não foram incluídas algumas das linguagens de classificação de vulnerabilidades e configurações existentes, permitindo assim, otimizar o tempo de desenvolvimento da aplicação para poder apresentar resultados válidos relativamente à deteção de vulnerabilidades em aplicações instaladas em sistemas operativos da família Windows.

Apesar dos resultados obtidos provarem que a aplicação cumpre com os seus principais objetivos, é justo afirmar que um projeto tecnológico nunca está realmente terminado, e assim sendo, esta aplicação apresenta alguns aspetos que devem ser melhorados em futuras versões.

Uma das funcionalidades da aplicação que deve ser melhorada é a deteção do software instalado numa determinada máquina. Esta funcionalidade encontra-se muito limitada neste momento e deve ser revista, para aumentar o número de informações sobre vulnerabilidades apresentadas ao utilizador quando realiza a deteção numa determinada máquina.

Os testes realizados para detetar vulnerabilidades também devem ser aprimorados pois tal como foi referido no ponto 5.3, o número de variações e variáveis existentes na linguagem OVAL para a realização de testes de conformidade, podem não estar completamente abrangidos ou apresentarem falhas não detetadas neste momento. Além deste aspeto, também deve ser adicionada uma nova funcionalidade na aplicação, para poder enviar os resultados, quando a aplicação perde a ligação ao servidor de forma inesperada. Ou seja, a aplicação deve realizar os testes em falta e após a conclusão, enviar os resultados para o servidor.

A janela com uma linha de comandos para o utilizador realizar a autenticação deverá ser alterada para uma interface gráfica simples, de forma a melhorar o aspeto da aplicação.

Relativamente à plataforma da Sparkint, apresentaram-se duas propostas para acrescentar mais informações associadas aos resultados obtidos após a execução do cliente. Uma delas consiste em adicionar informações sobre fraquezas (CWE) associadas às vulnerabilidades detetadas, aumentando assim a informação fornecida

ao utilizador. A outra proposta, está relacionada com a adição de informações de mitigação sobre as vulnerabilidades detetadas.

Pondera-se, por fim, a possibilidade desta aplicação vir a detetar vulnerabilidades noutros sistemas operativos, nomeadamente na família Unix, no entanto a prioridade será melhorar a aplicação já existente, com o fim de vir a ser comercializada, mantendo o foco nos pontos acima mencionados. Tudo irá depender da aceitação e interesse dos potenciais clientes.

Em suma, ficou provado que a aplicação desenvolvida consegue detetar vulnerabilidades em aplicações instaladas, em sistemas operativos da família Windows. Todos os objetivos propostos no início deste estágio foram atingidos com sucesso, constituindo uma boa base para o desenvolvimento futuro da aplicação. Tendo em conta que esta aplicação foi desenvolvida através de ferramentas de utilização livre, a sua comercialização poderá apresentar um valor relativamente mais baixo comparando com as aplicações já existentes no mercado. Espero que esta aplicação venha a ser bastante útil, na sensibilização dos utilizadores e organizações que decidam utilizar a mesma, tirando partido da mesma para reunir mais informações sobre vulnerabilidade e ataques a sistemas informáticos podendo assim realizar tarefas de mitigação e prevenção mais eficazes.

Bibliografia

[1] Peterson, T. F., and Eric Bender. 2011. "Nightwork: A History of Hacks and Pranks at MIT". MIT Press. (citado na pág. 7)

[2] RFC2828 (2010). Glossário de Segurança de Internet (RFC 2828). (citado na pág. 10)

[3] MITRE Corporation. "Common Weakness Enumeration". [Online]. Avaliable: `https://cwe.mitre.org/`. [Acedido em: Maio/2016]. (citado na pág. 11)

[4] MITRE Corporation. "Common Vulnerability Enumeration". [Online]. Avaliable: `https://cve.mitre.org/`. [Acedido em: Maio/2016]. (citado na pág. 12)

[5] Forum of Incident Response and Security Teams. "Common Vulnerability Scoring System". [Online]. Avaliable: `https://www.first.org/cvss`. [Acedido em: Maio/2016] (citado na pág. 12)

[6] Seth Hanford. 2015. "CVSS v3 Hands-on Training". In 27th Annual FIRST Conference on Computer Security Incident Handling. Berlin. (citado na pág. 12)

[7] Forum of Incident Response and Security Teams. "Common Vulnerability Scoring System - User Guide". [Online]. Avaliable: `https://www.first.org/cvss/user-guide`. [Acedido em: Maio/2016] (citado na pág. 12)

[8] MITRE Corporation. "Common Platform Enumeration". [Online]. Avaliable: `https://cpe.mitre.org/`. [Acedido em: Maio/2016]. (citado na pág. 13)

[9] Waltermire, D., Cichonski, P. and Scarfone, K. 2011. "Common Platform Enumeration: Applicability Language Specification Version 2.3 - NIST Interagency Report 7698". National Institute of Standards and Technology,

Gaithersburg MD, 20899. DOI=http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf (citado na pág. 13)

[10] National Institute of Standards and Technology. "Common Configuration Enumeration". [Online]. Avaliable: https://nvd.nist.gov/cce.cfm. [Acedido em: Maio/2016] (citado na pág. 13)

[11] MITRE Corporation. "Common Configuration Enumeration". [Online]. Avaliable: https://cce.mitre.org/. [Acedido em: Maio/2016] (citado na pág. 13)

[12] Center for Internet Security. "Open Vulnerability and Assessment Language". [Online]. Avaliable: https://oval.cisecurity.org/. [Acedido em: Maio/2016] (citado na pág. 14)

[13] Munyan, B. 2015. "OVAL Repository Transition". Center for Internet Security. DOI=http://csrc.nist.gov/news_events/cif_2015/security-automation/day3_security-automation_1035-1125.pdf (citado na pág. 14)

[14] National Institute of Standards and Technology. "Security Content Automation Protocol". [Online]. Avaliable: https://scap.nist.gov/. [Acedido em: Maio/2016] (citado na pág. 15)

[15] Quinn, S., Scarfone, K., Barrett, M. and Johnson, C. 2010. "Guide to Adopting and Using the Security Content Automation Protocol (SCAP)Version 1.0 - Special Publication 800-117 ". National Institute of Standards and Technology, Gaithersburg MD, 20899. DOI=http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-117.pdf (citado na pág. 15)

[16] National Institute of Standards and Technology. "National Vulnerability Database". [Online]. Avaliable: https://nvd.nist.gov/. [Acedido em: Maio/2016] (citado na pág. 16)

[17] Silva, Rui. 2015. "Todos os Nomes - Sistemas de Classificação de Segurança de Dados". Em VI Simpósio de Segurança Informática e Cibercrime - SimSIC 2015. Beja (citado na pág. 16)

[18] GFI Software. "GFI Languard". [Online]. Avaliable: http://www.gfi.com/products-and-solutions/network-security-solutions/gfi-languard. [Acedido em: Maio/2016] (citado na pág. 19)

[19] Tenable Network Security. "Nessus Vulnerability Scanner". [Online]. Avaliable: http://www.tenable.com/products/nessus-vulnerability-scanner. [Acedido em: Maio/2016] (citado na pág. 20)

[20] Qualys. "Qualys Vulnerability Management". [Online]. Avaliable: https://www.qualys.com/suite/vulnerability-management/. [Acedido em: Maio/2016] (citado na pág. 21)

[21] OpenVAS. "The Open Vulnerability Assessment System". [Online]. Avaliable: http://www.openvas.org/. [Acedido em: Maio/2016] (citado na pág. 22)

[22] Joseph D. Touch. 1995. "Performance analysis of MD5". In Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '95), David Oran (Ed.). ACM, New York, NY, USA, 77-86. DOI=http://dx.doi.org/10.1145/217382.217414 (citado na pág. 28)

[23] MITRE Corporation. "OVAL Documentation". [Online]. Avaliable: http://ovalproject.github.io/getting-started/tutorial/. [Acedido em: Junho/2016] (citado na pág. 39)